1993年4月1日発行（毎月1回1日発行）第12巻4号通巻132号　昭和58年11月2日第三種郵便物認可

特集 X68第7世代へ

X68030の目指す世界とソフトウェア/MC68030の概要

決定！ 1992年GAME OF THE YEAR/名作ゲーム再遊記

第5回アマチュアCGAコンテスト入選作品発表

4
1993

SOFT BANK
オー／エックス
定価600円

SHARP
目の付けどころが、
シャープでしょ。
夢の、頂きへ。
68ワールドの最高峰。
X68030
32bit PERSONAL WORKSTATION

32ビットパーソナルワークステーション

演算速度4.3倍(当社10MHz機比)/2.4倍(当社XVI比)※1、動画ウィンドウに見る新創造次元。選ばれた人だけが持つ感性によってX68030の扉はひらかれる。

X68000シリーズとして初の32ビットMPU MC68EC030を搭載し て高速化を実現。

データキャッシュ、プログラムキャッシュをそれぞれ256バイト搭載したクロック周波数25MHzの高速32ビットMPUを搭載。演算速度は2倍以上(当社従来比)※1の高速化を実現しました。また数値演算プロセッサMC68882※2(25MHz)もサポート。大量の実数演算を必要とするクリエイティブワークやGUI環境の操作性など、実行速度の飛躍的な向上が図られています。

※1 Dhrystn(四則演算)比。25MHz・データキャッシュオン・プログラムキャッシュオンでMC68000/10MHz時の約4.3倍、16MHz時の約2.4倍。

※2 数値演算プロセッサCZ-5MP1標準価格54,800円(税別 3月25日発売予定):本体内の専用ソケットに取りつけ可能。

65,536色表示、動画表示を実現。さらにパワーアップしたSX-WINDOWver.3.0。

X68000独自の本格的ウィンドウシステムとして定評の「SX-WINDOWver.2.0」をさらに強化した「SX-WINDOWver.3.0」を標準装備。新たに、65,536色の自然色グラフィック表示を可能とした『グラフィックウィンドウ』※を搭載。またアニメーション動画をウィンドウ上で表現でき、手軽にコンピュータアニメーションが楽しめる『CGAウィンドウ』、さらに従来のエディタのイメージを一新、高度な日本語文書作成をサポートするSX-WINDOW対応の高機能日本語マルチフォントエディタを標準装備。アウトラインフォントの展開もさらに高速化が図られています。

※SX-WINDOW上の512×512ドットのエリア内で表示可能。

GUIに対応する大容量メインメモリを搭載。

メインメモリは標準で4Mバイト、複数のアプリケーションをウィンドウ上で同時に使用するなど大量のデータ処理に対応。また本体内の増設で、I/Oスロットを使用せず最大12Mバイトまで拡張できます。拡張したメモリはすべて32ビットバスによる高速アクセスが可能、優れた拡張環境でシステムパワーアップをサポートします。

※メモリ増設には、4MB内部増設メモリボードCZ-5BE4標準価格54,800円(税別・3月25日発売予定)、4MB増設メモリモジュールCZ-5ME4標準価格49,800円(税別・3月25日発売予定)をご使用ください。なおCZ-5ME4はCZ-5BE4上に装着します。

X68000シリーズの高機能を継承した上で、さらに使いやすさの向上を図ったコンパチビリティ重視設計※1、すぐに使える高機能ソフトを標準装備。

●25MHzでは速すぎるアプリケーションも、従来のクロック周波数(10MHz/16MHz)で動作可能なソフトコンパチ重視設計●65,536色同時発色の自然色グラフィックス(最大表示エリア512×512ドット)、1024×1024ドットの実画面エリアを持つ高解像度表示能力(最大表示エリア768×512ドット・カラー液晶ディスプレイ使用時※2は640×480ドット)、疑似高解像度スーパーインポーズ(インターレース方式/512×512ドット・専用ディスプレイテレビ使用時)を装備した高精細度自然色グラフィックス機能。●外部MIDI音源もコントロール可能※3、ウィンドウ上で手軽にコンピュータミュージックが楽しめるMIDI音源対応デバイスドライバ搭載●ステレオ8オクターブ8重和音FM音源、ADPCM搭載●プリンタ、RS-232C、SCSI、オーディオ入出力、イメージ入力など多彩なインターフェイスを装備。●日本語変換効率や操作性を高めた日本語フロントプロセッサASKver3.0搭載。●従来のエディタのイメージを一新したSX-WINDOW対応の高速多機能日本語マルチフォントエディタ標準装備●日本語マルチフォントエディタ中に貼り付ける絵やグラフなどが簡単に作成できるグラフィックパターンエディタ●MIDI対応のX-BASIC。

※1 アプリケーションソフトおよび周辺機器のうち、一部動作しないものがあります。詳しくはシャープお客様相談窓口にお問い合わせください。

※2 10.4型カラー液晶ディスプレイLC-10C1-H標準価格598,000円(税別)、接続ケーブルAN-1515X標準価格4,200円(税別)をご使用ください(SX-WINDOW対応アプリケーションのみ、色数に制限があります)。

※3 別売のMIDIインターフェイスが必要です。

5.25"FDDマンハッタンシェイプシリーズ

■X68000伝統のマンハッタンシェイプを継承 ■5.25インチFDD2基搭載 ■80MBハードディスク内蔵(CZ-510C)※ ■マウス・トラックボール標準装備 ■ASCII準拠フルキーボード採用

※CZ-500Cには、2.5インチ80MB内蔵用ハードディスクドライブCZ-5H08(5月発売予定)/2.5インチ160MB内蔵用ハードディスクドライブCZ-5H16(6月発売予定)を用意しています。

X68030
32bit PERSONAL WORKSTATION

本体+キーボード+マウス・トラックボール
5.25インチFDDタイプ CZ-500C-B(チタンブラック)標準価格398,000円(税別)
HDDタイプ CZ-510C-B(チタンブラック)標準価格488,000円(税別)
14型カラーディスプレイ
CZ-608D-B(チタンブラック)標準価格94,800円(税別・チルトスタンド同梱)

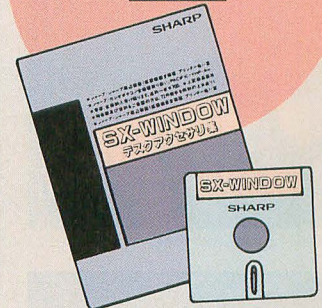
3.5"FDDコンパクトシリーズ

■32ビットのハイパワーを凝縮したコンパクトフォルム ■3.5インチFDD2基搭載 ■80MBハードディスク内蔵(CZ-310C)※ ■マウス標準装備 ■コンパクトキーボード採用

※CZ-300Cには、2.5インチ80MB内蔵用ハードディスクドライブCZ-5H08(5月発売予定)/2.5インチ160MB内蔵用ハードディスクドライブCZ-5H16(6月発売予定)を用意しています。

本体+キーボード+マウス
3.5インチFDDタイプ CZ-300C-B(チタンブラック)標準価格388,000円(税別)5月14日発売予定
HDDタイプ CZ-310C-B(チタンブラック)標準価格478,000円(税別)5月14日発売予定
14型カラーディスプレイ
CZ-608D-B(チタンブラック)標準価格94,800円(税別・チルトスタンド同梱)

デビュー

春一番
プレゼントセール
実施中!

X68030シリーズを
お買い上げいただき、
EXEクラブにご入会の方
先着1,000名様に

〔EXEクラブ入会申し込みハガキは本体同梱〕
既入会の方も、X68030の新会員NO.で新規登録します。

SX-WINDOWデスクアクセサリ集 内容

■SX-WINDOWの環境をより良くするツール
- ウィンドウアイコニファイ
- ファイルサーチ
- キーノート
- スクラップブック
- ソフトウェアキーボード
- フォントリンカ

■楽しめるツール
- スクリーンセーバ
- ミュージックボックス
- パズル

■実用的なツール
- 電子手帳通信ツール(DB-Z、PV-F1に対応)
- アドレス
- スケジューラ

■お問い合わせは… シャープ株式会社 コンシューマーセンター 西日本相談室 〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表)
電子機器事業本部システム機器営業部 〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表)

資料請求券 X68030 Oh!X 4月

特集　X68第7世代へ

第5回アマチュアCGAコンテスト

スターフォース

蒼き狼と白き牝鹿・元朝秘史

決定! 1992年GAME OF THE YEAR

特別企画　名作ゲーム再遊記

Oh!X

C O N T

●特集

〈スタッフ〉
●編集長／前田　徹　●副編集長／植木章夫　●編集／浅井研二　山田純二　豊浦史子　●協力／有田隆也　中森　章　林　一樹　吉田幸一　華門真人　吉田賢司　影山裕昭　大和　哲　村田敏幸　丹　明彦　三沢和彦　長沢淳博　宮島　靖　司馬　護　浦川博之　石上達也　柴田　淳　御木徳高　瀧　康史　●カメラ／杉山和美　●イラスト／山田晴久　寺尾響子　高橋哲史　川原由唯　●アートディレクター／島村勝頼　●レイアウト／元木昌子　ADGREEN　●校正／グループごじら

表紙絵：須藤 牧人

E　N　T　S

1993 APR.
4

■広告目次

SHARP

“感性”咲かせるワー

POWER WORKSTATION

インテリジェントなパフォーマンスを誇るX68000Compact XVIと
多彩にラインアップされたペリフェラル。感性を刺激するクリエイティブな
ワークステーション環境が自在に構築できます。

- パーソナルワークステーション(2HD3.5インチFDDタイプ・本体+キーボード+マウス)
  CZ-674C-H(グレー) 標準価格**298,000**円(税別)
- 15型カラーディスプレイテレビ
  CZ-614D-TN(チタンブラック)・**-BK**(ブラック) 標準価格**135,000**円(税別)
  ■ディスプレイテレビ/CZ-6TU用RGBケーブル**CZ-6CR1** 標準価格**4,500**円(税別)
  ■ディスプレイテレビ/CZ-6TU用TVコントロールケーブル**CZ-6CT1** 標準価格**5,500**円(税別)
- 80MB内蔵用ハードディスクドライブ
  CZ-68HA 好評発売中
- 5.25インチ増設用フロッピーディスクドライブ
  CZ-6FD5 標準価格**99,800**円(税別・接続ケーブル同梱)
- 光磁気ディスクユニット
  CZ-6MO1 標準価格**450,000**円(税別)
  ■SCSI変換ケーブル**CZ-6CS1** 標準価格**12,000**円(税別)
- 2MB増設RAMボード
  CZ-6BE2D 標準価格**54,800**円(税別・取り付け費別)
  ■2MB増設RAM**CZ-6BE2B** 標準価格**54,800**円(税別・取り付け費別)×2
  ■数値演算プロセッサ**CZ-6BP2** 標準価格**45,800**円(税別・取り付け費別)
- 48ドット熱転写カラー漢字プリンタ
  CZ-8PC5-BK(ブラック) 標準価格**96,800**円(税別)
- MIDIボード
  CZ-6BM1A 標準価格**26,800**円(税別)
- インテリジェントコントローラ
  CZ-8NJ2 標準価格**23,800**円(税別)

(68買ったら EXEクラブへ 入ろう!)

EXEクラブって何だ?

X68000を手に入れたら、やっぱり他のユーザーがどんな風に使っているのか気になるもの。ということでEXEクラブは、そんなあなたのための、他の68ユーザーとのコミュニケーションをバックアップする、情報交換の場です。

資料請求券
X68000ペリフェラル
Oh!X
4係

GRAPHIC WORKSTATION

- パーソナルワークステーション(2HD3.5インチFDDタイプ・本体＋キーボード＋マウス)
  CZ-674C-H(グレー) 標準価格**298,000**円(税別)
- 21型カラーディスプレイ **CU-21HD** 標準価格**148,000**円(税別)
- 80MB内蔵用ハードディスクドライブ **CZ-68HA** 好評発売中
- 光磁気ディスクユニット **CZ-6MO1** 標準価格**450,000**円(税別)
  ■SCSI変換ケーブル**CZ-6CS1** 標準価格**12,000**円(税別)
- 2MB増設RAMボード **CZ-6BE2D** 標準価格**54,800**円(税別・取り付け費別)
  ■2MB増設RAM**CZ-6BE2B** 標準価格 **54,800**円(税別・取り付け費別)×2
  ■数値演算プロセッサ**CZ-6BP2** 標準価格**45,800**円(税別・取り付け費別)
- カラーイメージスキャナ
  CZ-8NS1 標準価格**188,000**円(税別)
  ■スキャナ用パラレルボード**CZ-6BN1** 標準価格**29,800**円(税別)
- カラーイメージジェット
  IO-735X-B(ブラック) 標準価格**248,000**円(税別)
  ■接続ケーブル**IO-73CX** 標準価格**5,500**円(税別)

STANDARD WORKSTATION

- パーソナルワークステーション
  (2HD3.5インチFDDタイプ・本体＋キーボード＋マウス)**CZ-674C-H**(グレー) 標準価格**298,000**円(税別)
- 14型カラーディスプレイ**CZ-608D-H**(グレー) 標準価格**94,800**円(税別)
- 5.25インチ増設用フロッピーディスクドライブ**CZ-6FD5** 標準価格**99,800**円(税別・接続ケーブル同梱)

TFT COLOR LCD WORKSTATION

- パーソナルワークステーション
  (2HD3.5インチFDDタイプ・本体＋キーボード＋マウス)**CZ-674C-H**(グレー) 標準価格**298,000**円(税別)
- 10.4型カラー液晶ディスプレイ**LC-10C1-H**(グレー) 標準価格**598,000**円(税別)
  ■接続ケーブル**AN-1515X** 標準価格**4,200**円(税別)

※カラー液晶ディスプレイを接続してご使用の場合、SX-WINDOW上のアプリケーション利用に限定されます。

本体同梱の入会申込ハガキを送るだけで、自動的に無料入会。さらに下記の特典付き。

メリット1	会員ナンバー入りオリジナル会員電卓がもらえる。
メリット2	各種フェアご優待・イベント案内等、数々の特典がある。

SHARP
SHARP
SHARP
COLOR IMAGE SCANNER
カラープリンタもスキャナも………
黒の統一美。
画像処理のベストマッチングシステム for X68000。
シャープ株式会社　〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表)

入力から、編集、出力、複写まで、カラーの世界は、シャープです。

▶INPUT

X68000用パラレルインタフェースを標準装備した高速コンパクト型イメージスキャナ。

カラーイメージスキャナ JX-220X……標準価格168,000円(税別)

●A4サイズの原稿を約50秒※1で高速読み取り●CCDセンサー採用。さらに中間調処理でシャープでリアルな画像を再現●ディザパターン指定機能※2や濃度補正機能※2など高度な画像処理機能で緻密な読み取りが可能●解像度200ドット/インチ(約7.9ドット/mm)。ズーム機能で1%きざみの拡大、縮小も可能●色ずれの少ない線順次(1走査)読み取り●X68000シリーズ用「スキャナツール」ソフトを標準装備●プリンタと直接接続することによりダイレクトプリント※3が可能●RS-232Cインタフェース/X68000シリーズ用専用パラレルインタフェースを標準装備。

※1:A4、2値出力、コンピュータへの実転送時間。
※2:表記機能はJX-220X本体使用であり、付属ユーティリティ使用時は異なります。
※3:別売のパラレルインタフェースケーブル(JX-22PC標準価格12,000円(税別))が必要です。

▶OUTPUT

3種類の制御コマンドモードを搭載。質感も鮮やかに再現する高品位カラーイメージジェット。

カラーイメージジェット IO-735X-B……標準価格248,000円(税別)

●シャープ独自のIOシリーズコマンド(Gモード)に加え、NM-9900モード(Nモード)、ESC/P24-84C準拠モード(Pモード)をサポート。一般文書の作成から、各種デザイン、建築用パースなどのCAD分野に対応●発色性に優れた普通紙対応の新黒インキ採用。専用紙はもちろんオフィスでよく使われる普通紙にも鮮明カラー印字●プリントバッファメモリ(128KB)の内蔵で、ホストコンピュータの拘束時間を軽減●48ノズル(各色12ノズル)採用の高速印字。A4-1ページを※約90秒でプリント(データ受信時間除く)●ビジネス用途に適したB4横用紙幅対応●OHPフィルム(専用)にも鮮明プリント●ノンインパクト方式ならではの静粛印字●インキ補充は簡単、経済的なカートリッジ方式

※261×174mm領域

IO-735X-B
対応アプリケーション

●SX-WINDOW対応ペイントツール
Easypaint SX-68K
CZ-263GW 標準価格12,800円(税別)

●WYSIWYGを実現、ドローグラフィックソフト
CANVAS PRO-68K
CZ-249GS 標準価格29,800円(税別)

●オリジナリティを活かせるポップアップツール
NEW Printshop PRO-68K ver2.0
CZ-221HS 標準価格20,000円(税別)

●マルチワープロ
Multiword PRO-68K
CZ-225BS 標準価格32,000円(税別)

●高速カード型リレーショナルデータベース
CARD PRO-68K ver2.0
CZ-253BS 標準価格29,800円(税別)

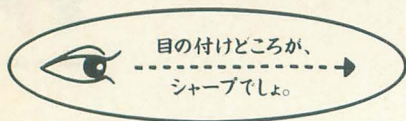
●パソコン通信もできるメモリ常駐型ソフト
Teleportion PRO-68K
CZ-258BS 標準価格22,800円(税別)

●これからの高速通信をサポート
Communication PRO-68K ver2.0
CZ-257CS 標準価格19,800円(税別)

資料のご請求・お問い合わせはコンシューマーセンター
●東日本相談室 〒261 千葉市美浜区中瀬1丁目9番2号 ☎(043)297-1221(大代表) ●西日本相談室 〒545 大阪市阿倍野区長池町22番22号 ☎(06)621-1221(大代表)

SHARP

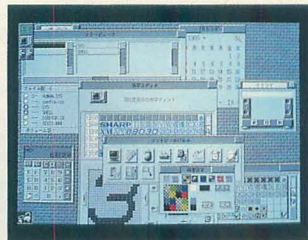
X68030/X68000シリーズ

成熟するウィンドウ環境で

65,536色対応、動画ウィンドウ標準装備。

SX-WINDOW ver3.0 システムキット

NEW

CZ-294SS(5インチ版)CZ-294SSC(3.5インチ版)3月発売予定

512×512ドットのエリア内で、自然描画に迫る美しい表現が可能な65,536色表示のグラフィックウィンドウを駆使できます。さらにグラフィックウィンドウ内でのアニメーション動画表示、各種グラフィックデータのコンバートも実現しました。高機能エディタ「日本語マルチフォントエディタ」を標準装備。アウトラインフォントの展開もフォントマネージャの効率化により、さらに高速化が図られています。その他、最大ズームサイズの設定や任意サイズのグラフィックを背景に設定できるなど、クリエイティブワークをサポートする数々の便利機能を装備しています。Human68k ver3.0およびASK68K ver3.0を標準装備しています。

※メインメモリ2MB以上必要です。※SX-WINDOW ver1.0/1.1/2.0をお持ちの方には有償バージョンアップを行います。

(日本語マルチフォントエディタの特長)

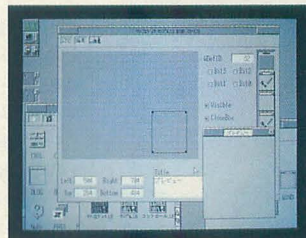
■自由なフォント設定: フォントタイプ、サイズ、スタイルを文字単位に指定可能。ルビも自由な大きさで付けられます。**■ワープロ機能:** 禁則処理(追い出し、ぶら下がりも指定可能)、ワードラップ(半角文字)。**■ユーザーカスタマイズ機能:** キー割り当て、マクロ定義、メニュー定義(アイコンも定義可能)、外部コマンドなど。**■イメージデータの貼り付け:** パターンエディタなどで作成したビットイメージデータの貼り付けが可能。**■シングルウィンドウモードの追加:** 複数のファイルをひとつのウィンドウで編集ができます。ファイルごとに編集環境の切り換えが可能。**■その他:** レイアウト機能の強化、矩形カット&コピー/矩形ペースト、マーク・ジャンプ機能。

待望のSX-WINDOW開発支援ツール。

SX-WINDOW 開発キット Workroom SX-68K

NEW

CZ-288LWD　3月発売予定

SX-WINDOW用のソフト開発に必要な開発ツールやサンプルプログラムを装備。プログラムの編集、リソースの作成、コンパイル、デバッグといった一連の作業をSX-WINDOW上で効率よく実行できます。初めてSX-WINDOW用のプログラムに挑戦する人にも、簡単に基本機能の理解ができる33種のサンプルプログラム付き。また各マネージャ解説と関数リファレンスの詳細なマニュアルも装備しています。 ※メインメモリ4MB以上、SX-WINDOW ver2.0以上、C compiler PRO-68K ver2.1が必要です。

キット構成

■開発ツール

●SXデバッガ
SX-WINDOW上で複数のプログラムを同時にデバッグすることができるソースコードデバッガ。

●リソースエディタ
SX-WINDOW上のリソースをリソースタイプごとの編集ウィンドウでビジュアルに作成・編集が可能。

●リソースリンカ
Cコンパイラやアセンブラで作成したリソースデータファイル(オブジェクトファイル)をリンクしてリソースファイルを作成。

●サンプルメイク
サンプルプログラムのコンパイル作業をSX-WINDOW上から、XCver2.1のMAKE.Xを呼び出して、自動実行する簡易メイクユーティリティ。

■サンプルプログラム

●基礎編(23種)
各マネージャの基本的な機能のみを用いた基本動作の理解。

●応用編(4種)
基礎編での基本機能を応用した簡単なアプリケーションの作成。

●実用編(6種)
基礎/応用編での機能を駆使した、実用的なアプリケーションの作成。

■その他ファイル

●インクルードファイル
Cコンパイラとアセンブラ用の関数定義、データ定義ファイル。

●ライブラリファイル
Cコンパイラ用関数ライブラリ。

マニュアル
●ユーザーズマニュアル ●プログラマーズマニュアル ●SXライブラリリファレンスマニュアル

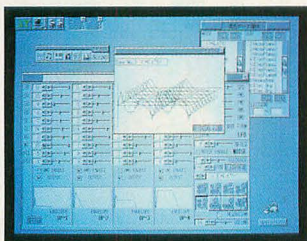
資料請求券
CZソフト
Oh!X
4係

さらに高度な創造次元へ。

●多彩なサウンドクリエイトを実現するFM音源サウンドエディタ。

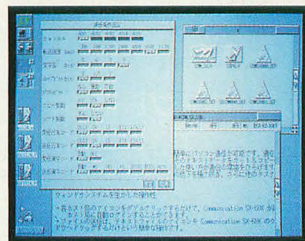
SOUND SX-68K

CZ-275MWD　標準価格15,800円(税別)

他のミュージックソフトで演奏中の音色を、簡単に作成、変更ができるマルチタスク機能、またエディット、イメージ、ウェーブの3つの編集/確認モードを装備。作成中の音色も50曲の自動演奏でリアルタイムに確認、編集できます。まさにミキサー感覚で音創りが楽しめるツールです。(2MB、ver1.1)

●マルチタスク機能をはじめ、通信環境がさらに充実。

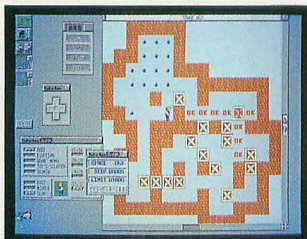
Communication SX-68K

CZ-272CWD　標準価格19,800円(税別)

通信環境をさらに高めたウィンドウ対応の通信ソフトです。マルチタスク機能により他のアプリケーションソフトを実行中でも簡単に通信が可能。また、ホスト局をクリックするだけの自動ログイン機能、初心者にも簡単なプログラム機能、最新モデム(20種類)もフルサポートしています。(2MB、ver1.1)

●SX-WINDOW対応の「倉庫番」がパワーアップ。

倉庫番リベンジ SX-68K ユーザー逆襲編 NEW

CZ-293AW(5インチ版)CZ-293AWC(3.5インチ版)4月発売予定

10年にわたるユーザーの投稿など、新作306面が目白押し。まさに倉庫番の最強版がSX-WINDOW上で楽しめます。移動可能先が表示されるAI機能を搭載、またマウスをクリックするだけで簡単に問題を作成できるエディット機能や、パターンを替えてちょっと違った雰囲気でゲームが楽しめるパターン変更機能も装備しています――。半年で解けたらあなたは天才？です。(2MB、ver1.1)

●ウィンドウ対応グラフィックツール。

Easypaint SX-68K

CZ-263GWD　標準12,800円(税別)

マウスによる簡単操作、65,536色中16色の多彩な表現、クリエイティブマインドに応えるウィンドウ対応ペイントツールです。同時に複数のウィンドウを開いて編集でき、各ウィンドウ間でのデータ交換もできます。(2MB、ver1.1)

●「SX-WINDOW開発キット」のサポートツール。

開発キット用ツール集

CZ-289TWD　開発中

SX-WINDOW開発キットをさらに使いやすくするためのツールです。SXコールの簡易リファレンスを簡単に検索する「インサイドSX」、イベントの発生を常時監視確認するイベントハンドラ、リアルタイムにメモリブロックの利用状況を表示するヒープビューアなど11種のツールが用意されています。(2MB、ver2.0)

※(2MB、ver1.1)の表示は、メインメモリ2MB以上、SX-WINDOW ver1.1以上が必要であることを示します。

充実の PRO-68K シリーズ

●マルチフォント印字に対応。

Multiword ver2.0 NEW

CZ-225BSV

標準価格32,000円(税別)

Zeit社の書体倶楽部をサポート。同時に6書体のフォントが指定可能、レーザプリンタのフォントも複数使用できます。またキー操作やメニューの改良、均等割り付け、グラフィックのアイコン化なども可能。

※MultiwordおよびMultiword ver1.1をお持ちの方には有償バージョンアップを行います。

●ビジネスグラフチャート。

CHART PRO-68K

CZ-267BSD　標準価格38,000円(税別)

※以上のPROシリーズのソフトの動作にはメインメモリ2MB必要です。

※発売予定のソフトの画面写真は実物とは異なる場合があります。

●お問い合わせは…シャープ㈱電子機器事業本部AVCシステム事業推進室　〒162　東京都新宿区市谷八幡町8番地　☎(03)3260-1161(大代表)へ　シャープ株式会社

X68000 Free Software Book

近刊

グループ68K 編
B5変形判・384ページ　5″2HDディスク付き
定価2,900円（本体2,816円　フロッピーディスク含む）

X68000ユーザだけど、パソコン通信はしたことがないという人のためのフリーソフトウェア入門書。通信ソフト、ファイラ、ビューワ、システム支援関係のツールなど、24本のフリーソフトウェアを収録し、その基本的な使用法を解説するとともに、パソコン通信のしかた、フリーソフトウェアの入手のしかたなども紹介。巻末には、フリーソフトウェアの作者による座談会「ぼくらは、なぜフリーソフトを作っているのか」を収めた、わかりやすく、読んで楽しいフリーソフトの入門書である。

●主な内容

第1章　X68000フリーソフトウェア入門
第2章　X68000フリーソフトウェア・セレクション
第3章　座談会「ぼくらは、なぜフリーソフトを作っているのか」
出席者：Ext(TwentyOne、dedit作者)
+YuNK(HIOCS+HAS作者)+西表山猫(TMN作者)
+星野美季(Telecom Miki作者)

■収録プログラム

●通信関係

TMN	ウィンドウを多用し、強力なマクロ言語を備えた高機能ターミナルプログラム
MuTerm	コンパクトで高速・高機能なターミナルソフト

●通信支援関係

ish	通信必携のテキストーバイナリ変換プログラム
LHA	高率圧縮を行う標準アーカイバソフト
Bdif & Bup	通信ユーザ必携のバイナリ差分抽出更新プログラム

●ファイラ

Fu	便利で使いやすく、自由度の高いファイルユーティリティ
MF	ユーザが独自の環境を構築できる、柔軟性に富んだ2画面型ファイラ

●ツール

LZX	.X、.Rファイルを実行可能なまま圧縮するツール
SEE	LZH(LHA)ファイルの内容も確認できるファイルビューワ
DC	超高速、連続複写可能なディスクコピーツール
SuperED	超高速、高機能なED.Xコンパチブルエディタ
tsort	数字を認識できるディレクトリsortコマンド
dedit	X68000ユーザ必携の多機能ディスクエディタ
SRAMCLR	内臓SRAMの内容を必要に応じて消去するプログラム

●システム関係

TwentyOne	パワーユーザ御用達のシステム強化プログラム
hcommand	TwentyOneなどのシステム拡張機能に対応させたCOMMAND.X
caps	プログラムごとに自由にキー割り当てを変更、拡張できる常駐プログラム
float2p	純正float2.X(Ver.2.01)をさらに高速化した浮動小数点演算ドライバ
HIOCS	高速文字表示が可能なコンソールドライバ
FLEXDISK	高速、再確保可能RAMディスクドライバ
DCACHE2	デバイスドライバに割り込んで動作するディスクキャッシュプログラム
DE	ドライブ名をデバイスごとに設定するツール
ADDDRV	デバイスドライバ用ユーティリティ
C/DINIT	起動時にCONFIG.SYSを選択することができるデバイスドライバ

4月上旬刊行予定

Outside X68000

桒野 雅彦 著　B5変形判　300ページ　予価5,000円

好評既刊書『Inside X68000』の外部拡張機器編。
本体編／周辺機器編／自作周辺機器編の3部構成で、X68000各機種の内部回路図をはじめ、主要拡張ボードのハードウェア情報を網羅。
X68000シリーズ向けに周辺機器の自作をめざす方の必備書。

SOFT BANK　発売
ソフトバンク出版事業部　〒108 東京都港区高輪2-19-13 NS高輪ビル TEL:03-5488-1360

安くて親切 TSUKUMO

4月18日(日)まで

フレッシュスタートセール

掲載商品2万円以上送料無料(一部地域を除く)

ツクモグローバルカード

大人気!入会者募集中! 18才以上なら学生さんもOK!

国内・外で活躍!使って便利、持ってて安心!ツクモグローバルカードはジャックス・VISAとの提携カードです。ツクモ各店でのお買物がらくらくできる上に、国内はもとより海外での分割ショッピングもOK!
20才以上の方にはキャッシングカードも発行致します。
お申し込みは☎03(3251)9898又は店頭にて!

シャープX68000の事なら何でも揃うツクモにおまかせ!

秋葉原を歩き回る必要はありません。情報が沢山。分らない事は何でもお尋ね下さい。目に優しい10.4型カラー液晶ディスプレイ(LC-10CI)も取り扱い中!詳しくはお問い合わせ下さい。システムのご相談は☎03(3253)1899までどうぞ。

X68030

FDDモデル CZ-500CB
HDDモデル CZ-510CB

ずっと待ち焦がれていた待望の新製品がついに発表!X68000シリーズ32ビット最上位機が…。

- 新たに32ビットCPU(MC68EC030/25MHz)を搭載し、従来機の3倍以上のスピードアップを実現!
- 成熟するウィンドウ環境、使いやすさと高機能を追求し、動画機能・SX-WINDOW Ver.3.0搭載
- SX-WINDOWの操作環境を考え、4MBメモリ内蔵
- 5インチ2HDフロッピーディスクドライブ2基搭載
- カラー液晶ディスプレイ接続可能

★おすすめの組み合せ★

CZ-500CB…………… ¥398,000
CZ-614DTN………… ¥135,000
RMO-S360…………… ¥218,000
SCSIケーブル………… サービス

発売記念ツクモ特価 お問い合わせ下さい

新製品発売のため旧製品大放出!お問い合わせ下さい

X68000用ドライブTSシリーズ大好評発売中!!

目のつけどころがツクモでしょ

●X68000シリーズ専用 3.5インチフロッピーディスクドライブ
TS-3XRシリーズ
〈仕様〉
- 3.5インチ2DD/2HD/2HCフォーマット対応
- ユーティリティソフト付属(デバイスドライバー/フォーマッター)
- 標準サイズケーブル付

TS-3XR1 定価¥44,800 1ドライブ ツクモ特価¥35,800
TS-3XR2 定価¥57,800 2ドライブ ツクモ特価¥46,800

●X68000 CompactXVIシリーズ用 5インチフロッピーディスクドライブ
TS-5XRシリーズ
〈仕様〉
- 5インチ2HD/2DDフォーマット対応
- ドライブ番号切り換えスイッチ付
- Compact XVI用ケーブル付

TS-5XR1 定価¥53,800 1ドライブ ツクモ特価¥42,800
TS-5XR2 定価¥72,800 2ドライブ ツクモ特価¥57,800

耳よりな情報—X68000XVI/Compact XVIシリーズをお持ちの方

2MB増設RAM (CZ-6BE2Bコンパチ)
TS-6BE2B
既に、内蔵メモリーボードを搭載して4MBに増設されている方で、更に増設をお考えの方へお勧め商品
特別限定生産 ツクモ特価¥34,800

おすすめSCSIタイプハードディスク

VIP-100CX (100MB ツクモ仕様) ツクモ特価¥59,800
VIP-120CX (120MB ツクモ仕様) ツクモ特価¥66,800
VIP-200CX (200MB ツクモ仕様) ツクモ特価¥86,800
LHD-B240HFM (240MB) ツクモ特価¥99,800
※SCSIボード(CZ-6BSI 定価¥29,800)は別売です。

X68000シリーズ用RAMボード

1MB増設RAMボード (CZ-600C専用) ツクモ特価¥19,500
1MB増設RAMボード (ACE/PRO/PRO2シリーズ用) ツクモ特価¥17,000
2MB増設RAMボード (拡張スロット専用) ツクモ特価¥33,800
4MB増設RAMボード (拡張スロット専用) ツクモ特価¥59,800
※計測技研のボードも取り扱い中!!お問い合わせ下さい。

さらに拡がるSX-WINDOWワールド

SX-WINDOW開発キット
CZ-288LWD 好評発売中
SX-WINDOW Ver3.0 近日発売
Communication SX-68K
CZ-272CWD ¥19,800
SOUND SX-68K
CZ-275MWD ¥19,800
Easypaint SX-68K
CZ-263GWD ¥12,800

パソコン通信時代は9600ボーへ!!

★モデム★
- 9600bps MNPクラス5 V.42bis搭載 ツクモ特価¥49,800より
- 14400bps MNPクラス5/V.42bis DSI製 SCOUT 144MX(ケーブル付) ツクモ特価¥69,800

★通信ソフト★
- たーみのる2 ツクモ特価¥13,000
- Communication SX-68K ツクモ特価¥16,800

大容量記憶装置

これが今一番の人気者!!更に安くスリムなボディの新製品登場
SONY 3.5インチ光磁気ディスクユニットセット
RMO-S360 ¥218,000
SCSIケーブルサービス
ツクモ特価販売中

MIDIコンピュータミュージック特選セット♬

その1
- CM-300 ¥58,000
- SX-68MII ¥19,800
- Mu-1 Super ¥39,800

合計定価¥117,600
ツクモ特価¥92,000

その2
- SC-55 ¥69,000
- SX-68MII ¥19,800
- Mu-1 Super ¥39,800

合計定価¥128,600
ツクモ特価¥99,000

その3
- CM-500 ¥115,000
- SX-68MII ¥19,800
- Mu-1 Super ¥39,800

合計定価¥174,600
ツクモ特価¥141,000

その4
- SC-33 ¥49,800
- SX-68MII ¥19,800
- Mu-1 Super ¥39,800

合計定価¥109,400
ツクモ特価¥88,000

夏のボーナス一括払(金利手数料なし)受付中!詳しくは各店までお問い合わせ下さい。

大好評 Matier—マチエール セット

その1 大ヒット発売中

WACOM製
- SD-510C タブレット…¥98,000
- TJ-410A-2 拡張ケーブル…¥6,000
- SP-200A スタイラスペン ¥10,000

サンワード
- Matier(マチエール)…………¥39,800

合計定価¥153,800
ツクモ特価¥128,000

その2

ヒューレットパッカード HP Desk Jet 505J インクジェットプリンタ¥99,800
カラーキット…………¥12,000
アーベル プリンタケーブル……¥4,800
サンワード Matier(マチエール)¥39,800
合計定価¥156,400
ツクモ特価¥123,000

その3

キャノン BJ-15V……………¥89,800
カットシートフィーダ…¥9,800
プリンタケーブル……¥4,800
カートリッジ…………¥15,000
(Y、M、C、それぞれ1個ずつ)
サンワード Matier(マチエール) ¥39,800
合計定価¥159,200
ツクモ特価¥123,000

通信販売のご注文は下記フリーダイヤルへ。
全国どこからでも通話料無料
受・注・専・用 フリーダイヤル **0120-377-999**
通販センター **03-3251-9911** 商品についてのお問い合わせは各店又は通販へ。

クレジット払い	各種リース払い	現金書留払い
月々¥3,000以上の均等払いも頭金なし、夏・冬ボーナス2回払いも受付中!	くわしくは各店にお問い合わせ下さい。ケースに合わせてご相談にのります!	〒101-91 東京都千代田区神田郵便局私書箱135号 ツクモ通販センター Oh!X係
カード払い(¥5,000以上)	**全国代金引き換え配達**	**銀行振込払い**
通信販売での御利用カード、ツクモグローバルカード、VIPカード、セントラル、ジャックス※御本人様より電話で通販部へお申し込み下さい。	お申し込みは☎03-3251-9911へお電話1本!配達日の指定もできます。	事前に☎でお届け先をご連絡下さい。三和銀行 秋葉原支店(普)1009939 ツクモデンキ

秋葉原各店 営業時間AM10:15〜PM7:00

ツクモパソコン本店2F
☎03-3253-1899(直通)(担当/荒井)
ツクモパソコン本店代表☎03-3253-5599 休毎週木曜日
ツクモニューセンター店 ☎03-3251-0987(担当 沢栄)休毎週木曜日
(下取り交換、中古販売も行っております。)
※定休日が祝日と重なる場合は営業致します

各古屋各店
名古屋1号店 ☎052-263-1655(担当/山尾) 営AM10:00〜PM7:00 休毎週火曜日
名古屋2号店 ☎052-251-3399(担当/松原) 営AM10:00〜PM7:00 休毎週水曜日

札幌各店
ツクモ札幌店 ☎011-241-2299(担当/田口) 営AM10:30〜PM7:30 休毎週木曜日
DEPOツクモ2番街店 ☎011-242-3199(担当/鈴木) 営平日AM10:40〜PM7:30 日・祝日AM10:10〜PM7:00 休毎週木曜日

ツクモは「スーパーX PRO SHOP」です。
PRO STAFF ツクモ
九十九電機株
〒101-91 東京都千代田区神田郵便局私書箱135号
★商品のご注文は在庫確認の上お願いします ★表示価格には消費税は含まれておりません

PCWEEK縮刷版

◆オールカラーの誌面構成と最近の記事内容

PC WEEK

省電力32ビットCPU「Hobbit」で AT&TがPDA市場での覇者狙う

ノベルがWFW統合を意識しLiteとDR DOS統合へ

AccessとFoxPro投入で MSがDB市場制覇を宣言

WORKSTATIONS

ノベルのUSL買収に関しUSLが米社長の来日記者会見を開催 中立性保持の意向とともに対マイクロソフトの戦略を強調

好調続きのWS市場にもついに不況の波来る！ 世界市場における'92年の収益は横ばいに終わる

アシストがAster*x日本語版に RDBMSのインタフェース搭載

SPECIAL REPORT

国内標準PC-9801と別れられない多くのユーザーのために

Tech View

一部分野で運用が始まり現実感を持ち始めた仮想現実の世界

今年の主な記事から

■ファイル共有から着実に浸透を始めたパソコンLAN

■低価格競争で市場規模が拡大!! 93年のDOS/V展望

■管理機能を付加するデータバックアップ

■日本IBMとキヤノンがNEC対抗のプリンター一体型の486ノートPC

■Pentium,PDA,低価格PCなどで93年米国パソコン市場は2桁成長

■ネットワーク業界の巨人ノベルがUSL取得でWindowsNTに対決姿勢

MULTIMEDIA

基礎技術は完成し普及期近づくマルチメディアPC

SOFTWARE

ジャストシステムが一太郎Ver.5でエプソンPC-386,486正式サポート

Windowsデスクトップを業務別に管理するシェル

レイアウト機能が改良されアイコン操作で操作性も向上したPageMaker QuarkXPressを意識した機能強化で市場での優位を堅固たるものに

SOFTWARE

Beta Sight

日本語入出力環境が整備され描画速度も向上するWindows 3.1J

WORKSTATIONS

USLがSVR4の日本語拡張機能のOEM開始 日本語入力システムの相互運用性を実現

Interview

HPはインフォメーションマネジメント時代の環境を提供する

■パソコン市場を巡る争いは「価格」から「価値」へ移行

■DOS6.0ツールはグラフィカルにMSが新バージョンの概要発表

■エプソンが486搭載で20万円台 NECも新機種でWin環境整備

■低価格DOS/V機に新生98の逆襲始まる! 主力シリーズはAT生まれの技術満載

■アップルが低価格68040マシン CentrisとQuadra新機種2月に

■サーバでもコンパック旋風!? SCSI-2搭載機が105万円から

HARDWARE

コンパックが機能,保守維持のまま低価格のデスクトップ機 デルやASTも値下げで拍車かかるATメーカーの価格競争

Top Interview

386マシンの性能を1.7倍にするサイリックス

FOCUS

Windowsの最先端技術を手軽に使えるVisual Basic

FOCUS

マルチメディア,GUIの先駆者ゲームメーカーの動向

SPECIAL REPORT

マルチベンダーの統合環境が進むMacのネットワーク

X68030

新たなる世代に向けて

32bit PERSONAL WORKSTATION

ようやく新世代X68シリーズの先陣を切ってX68030の5インチタイプが発売された。読者の方でもすでに手にしている方もいるかもしれない。ここではおいそれと見ることができない中身をどーんと公開してみよう。また，従来機でも使える新システムとパワーアップしたSX-WINDOW関係も詳しく紹介したい。

ついに発売されたX68030。今回は先月の速報で紹介しきれなかった細部についても触れてみたい。

発売されたのは5インチFDD内蔵モデルのCZ-500C/510Cだ(3.5インチモデルCZ-300C/310Cは5月に発売予定)。伝統のツインタワー，マンハッタンシェイプにチタンブラックと，外見上はX68000XVIとほとんど変わらない。そんななかでひときわ目を引くのが，真紅の「X68030」エンブレムだ。真紅の文字はX68000シリーズのハイエンドモデルであることを象徴しているのだそうだ。確か色彩心理学で黒に赤は力の象徴という解釈があったように思う。

だいたいの感触として，シャープからの公称値，

X68000の4倍強
X68000XVIの2倍強

というのはかなりあてにできる目安だと思っていい。NECが公称するPC-9801の100倍以上の性能アップよりは信頼できる数字だ。コスト的に考えるなら，もう少し手を抜いてもいいんじゃないかと思うくらいきっちり作られた68030マシンといえる。

「68030の25MHz」は正しく「25MHzの68030が32ビットバスに，メモリは同期バス(基本バスサイクル2クロック)でほぼノーウェイトで動作するマシン」を意味する。

ちなみにDhrystone値は68000用にコンパイルされたコードで5800程度になる。Dhrystoneベースでいけば，PC-9801VM相当のマシンからPC-9801DA相当になったということになる。

といっても，従来機からの買い換えを考えている人にとって(読者の大半だと思うが)は他機種との比較はあまり意味がないだろう。機種が違う場合，ベンチマークテストの結果は性能指標にはならないからだ。PC-9801VMがCPUだけで標準の3倍の大きさの画面のWINDOWSを動かしたらどんな具合になるか想像してみてほしい。

ハードウェアを見る

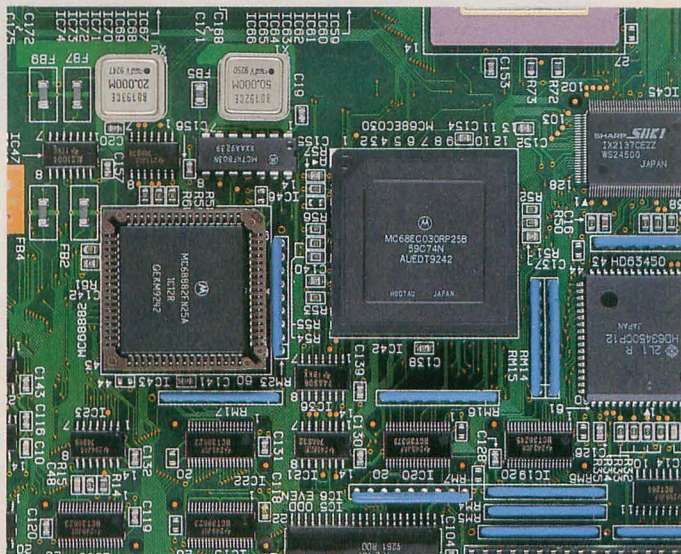
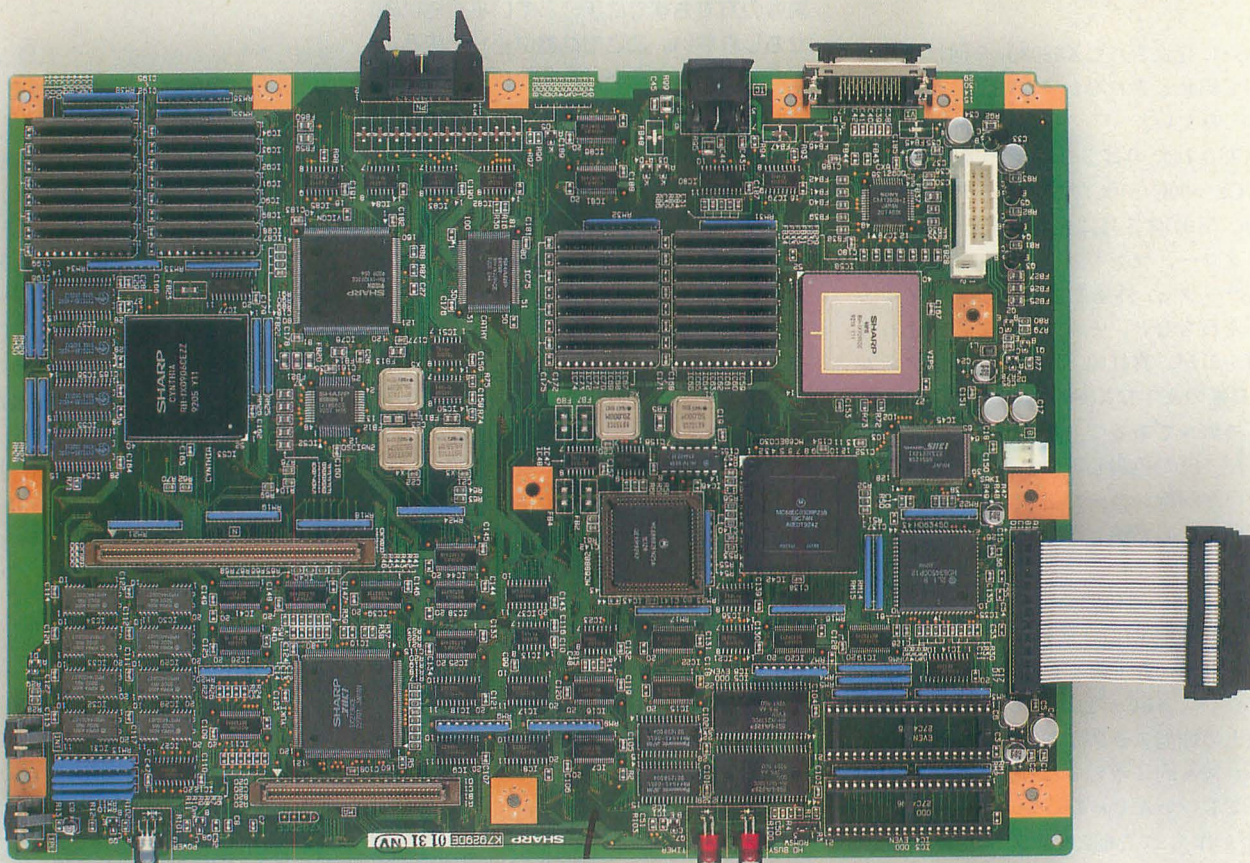
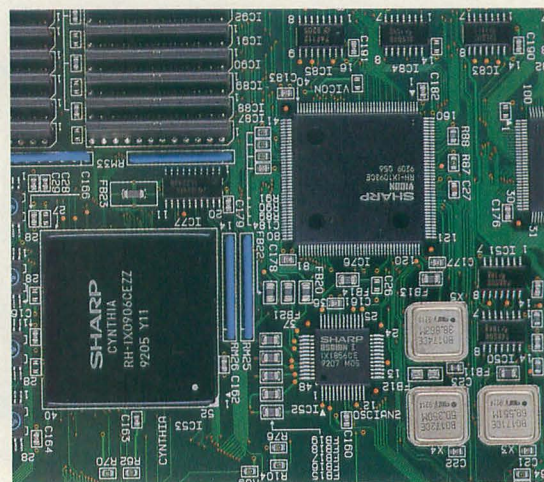
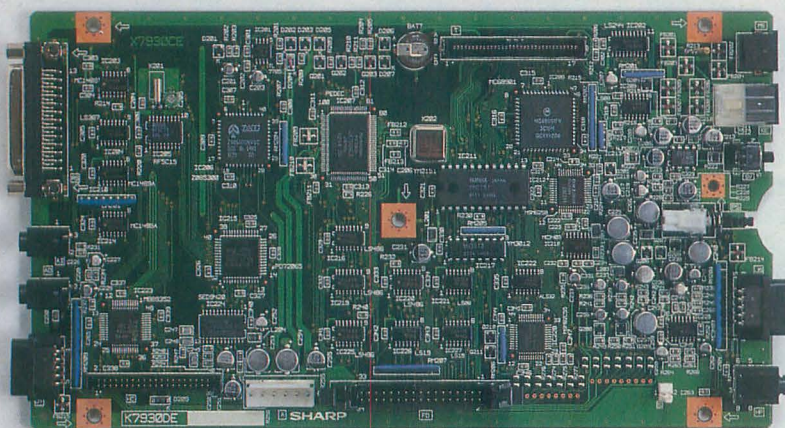
4層基板にびっしりと部品が配置され，いつもながら綺麗な仕上がりを見せている。バスが32ビットになることにより配線量は格段に増えているはずだが，基板のサイズは従来と変わらない。いまどきジャンパ線がないことなど当たり前だと思っていたら他機種ではそうでもないらしい。

3.5インチタイプはさらに小さな基板でまったく同じ内容なのだから，これはもう驚異だ。CompactXVIが登場したときも限界の大きさだと感じたものだが，これならノート型も不可能ではなさそうに思える。

初代機からお馴染みの石はCYNTHIAだけとなっている。LSIの基本的な構成はX68000XVIとあまり変わっていない。DMAは12.5MHzの高速型が採用されているが，データ転送は，もはやCPUを使ったほうが高速に行える。ちなみに63450ではこれ以上高速な製品は存在しない。ディスクやAD PCMの制御回路の一部といった感じだ。

バス幅の32ビット化で登場したのが，メモリコントローラYUKIだ(もうひとつ小さなSAKIというのもある)。左上の大きな新カスタムチップが見えるのがそれだ。ちなみに歴代のメモリコントローラを並べるとET，OHM，ASA，YUKIとなる。

同一ページ内ノーウェイト(4Kバイト/ページ)で動作し，そうでない場合は2クロックを要する。「ほぼノーウェイト」というのは正しい。ノーウェイトが基本だから，2次キャッシュというものは存在しない。

I/Oは16ビットバスで接続されている。周辺はX68000のバスを踏襲しているので，周辺機器は従来どおり10MHzで動いている。とはいっても，プログラム自体の速度差が顕著に表れてくるので，多少の調整はしてあるようだ。X68000XVIで問題になっ

X68030のCPU周辺。68882はQFPパッケージとなった

ていたFM音源へのアクセスなどもハードウェアでウェイトを入れてくれるように改善されている。しかし外部に接続されるMIDIボードのアクセスなどではプログラム側でウェイトを入れなければならなくなったようだ（25MHz，キャッシュ動作時）。I/O関係はタイミング重視なので，単に速くなればいいというものではないのだ。

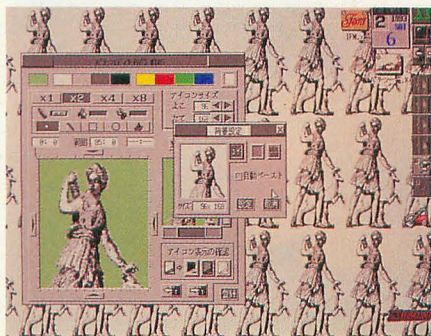
ここでは写真が挙げられていないが，ハードディスクはツインタワーモデルでも2.5インチタイプが採用されている。確かに絶対的な性能はなかなか凄いのだがコストパフォーマンスでいうと3.5インチドライブのほうが圧倒的に高い。また，2.5インチでは大容量のドライブはまだほとんどない。最近200Mを超えるものが現れたばかりで，X68030シリーズの160Mバイトというのが現行の限界と考えていい。

左側タワーを開けてみるとスペース的には従来と変わらず，3.5インチドライブが入りそうなスペースが確保されている。2.5インチドライブなら2ドライブ入りそうな感じだ。マウント用の金具と接続ケーブルをなんとかすれば3.5インチ大容量ドライブ

左上はCZ-500Cの底面に位置するサブ基板。CZ-300Cではこの部分もまとめてメインボードが構成されている。右上はスプライトコントローラCYNTHIA周辺の拡大図。X68シリーズのもうひとつの「顔」ともいえる。そしてメイン基板。左下にメモリコントローラYUKI，CPUの右手にSAKIがある。YUKIの下のコネクタに増設メモリが接続される。右上の目立つ石はVIPS。X68030のやたら複雑な画面合成などを制御するビデオコントローラだ。ちなみにCRTCのVICONはCYNTHIAの横の大きいやつ。右下には2つ並んだROMと新しいシステムに変えるときに使うROMソケットだ。見てのとおりインテルは入っていない（8255は71055になっている）。

コピー時に確認がついた

背景の描き換えも自在

アクセサリ集も発売される

を積むことも不可能ではあるまい。

ソフトウェア編

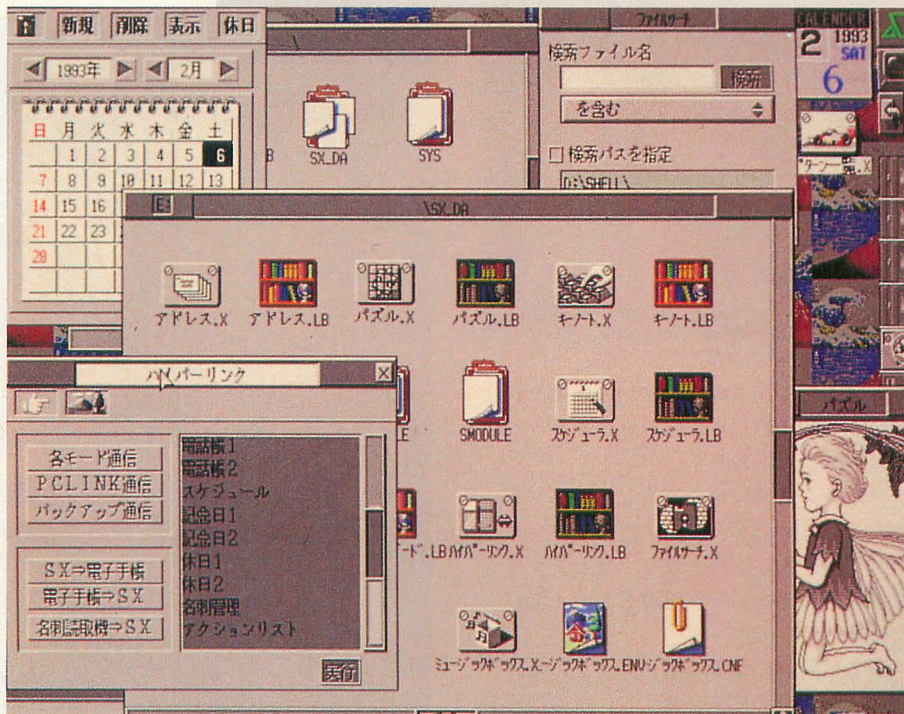
電脳倶楽部でX68030の可動ソフトリストが掲載されていたのだが，シャープ提供の資料とDōGAでの独自調査したものと編集部で独自調査したもので微妙な違いがある。試作機レベルではまだ評価段階にはないのかもしれない。ということで，具体的な動作チェックリストについては来月以降，発売版のマシンとシステムが揃った時点で開始する。

現状で動かないソフトは3割くらいあると思われるが，ハードだけでは吸収できない原因というのはそう多くはない。傾向別に対策を立てることでほとんどのソフトが動作するようになると期待される。

今後は個別対応で動かないものを動かしていく予定である。

ウィンドウ環境の充実

新世代ソフトウェアの柱はSX-WINDOW ver.3.0である。

X68030の登場によってSX-WINDOWが快適な速度で動作するようになった。CPUの高速化がもっともうれしいのはグラフィックやウィンドウ環境で使う場合だ。ゲームなんかは速くなってもしかたがない（ちゃんと作られたものなら，だが）。その意味では，X68030はSX-WINDOWマシンという位置づけもできる。

SX-WINDOW ver.3.0は各部がやや高速化されているので従来機で使用する場合にも魅力的なソフトウェアである（さすがに10MHzのマシンでは後ろで動画を動かしたりしてはいけないが）。

もともと極端に遅いウィンドウシステムではなくなっているので10MHzのマシンでもウィンドウ再描画などでワンテンポ待つのを我慢すれば一応使える。16MHzで動かすのなら実用レベルといえるだろう。ストレスなく使いたいという場合はX68030を導入すべきだろう。

これまでSX-WINDOWはそのポテンシャルをまるで発揮できていなかった。

誰もが望んだマルチタスク/マルチウィンドウ環境であり，そこには未知の可能性が広がっているとわかっているのに，移行することができなかった人が多いのだ。SX-WINDOWの速度は「より快適な環境を作りたい」という欲求による選択には少々，分が悪かった。

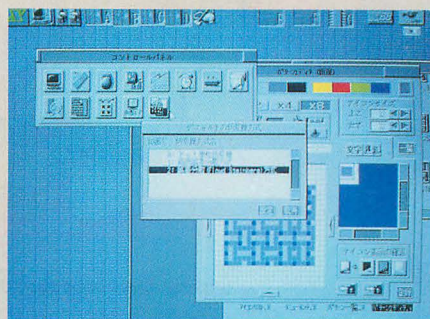
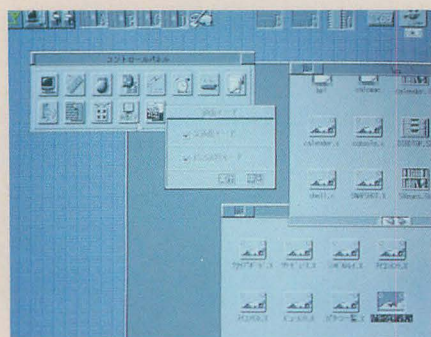
新機種，新システム，そして開発ツールの発売とともに，その分のフラストレーションからの反動が一気に爆発しそうな予感がある。ウィンドウの時代は間近いかもしれない。

65536色の可能性

ひとつの転機といえるのが，SX-WINDOWの多色化だ。グラフィック環境は，これまでの16色グラフィックを使ったモードと65536色が表示できるモードに2分されることになる。従来のEasypaintSX-68Kなどは65536色モードでは動作しないし，65536色対応のアプリケーションも16色モードで動作させることはできない（でも色変換のマネージャがあるんだから……）。今後，Easypaintに代わるものは現れるかもしれないが，SimEarth，SimAntが16色なのは痛い。

この2つのモードを切り替えるためにはSXシェルの再起動が必要になるのだが，このあたりは改善が期待される。

おそらく今後は65536色モードのほうが主流になっていくと思われる。表示面積さえ考えなければ理論上65536色モードは16色モードを吸収できるはずなのだ。

現在は画像ファイルのビュアーと動画のビュアーしか用意されていないが，2D/3Dのグラフィックツールがまず期待される。レイトレーシングなんかバックグラウンド

色変換マネージャが装備された

まず，画面モードを設定する

で計算させて，表画面では別の作業を行うというのが理想だろう（やっぱりトランスピュータかな？）。

マルチタスクだから適当なインタフェイスを用意すればZ's-EXのようなプラグインフィルタという概念はなくなる。マルチタスクはグラフィック環境をも変えていくだろう。

ここで問題になるのがメモリである。グラフィックがメモリを食うというのは容易に理解してもらえると思う。ちょっとPIC画像を……と思っても512Kバイトずつ消費されていくのだから。

とりあえず高解像度，1：1に近いドット比，65536色というパソコンでは最高峰といえるグラフィック環境が用意されたわけだ。これでまた使い甲斐があるウィンドウシステムになったといえる。

もっと解像度を

ウィンドウ環境が主体になってくるとディスプレイ解像度が改善されなかったことで不満も出てくる。どうせなら従来機種でも使える高解像度対応のディスプレイなり，周辺機器なりを発売してほしくなる。

ちなみに，

SXWIN -G90

という風にオプションをつけて起動すると24kHzインタレースモードを使った1024×848ドットモードとなる（65536色使用可）。画面のちらつきが大きいため長残光タイプのディスプレイでもないと実使用には向かないが，ウィンドウ操作には快適な広さが得られる。

元に戻すときは，

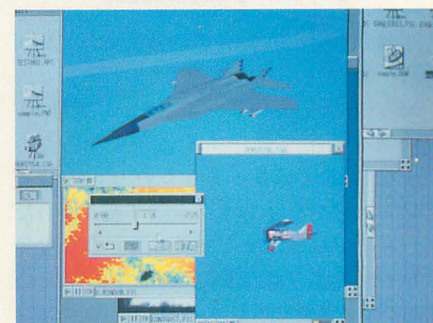
SXWIN -G88

だ。このように現状でも高解像度表示が可能なソフトウェアとハードウェアを標準で持ちながら，使わないのはもったいない話である。少し改良すればちらつきのない高解像度ディスプレイに接続することもでき

16bit Color

TIFFやJPEGといった標準的な画像フォーマットに対応したキャンバス．X

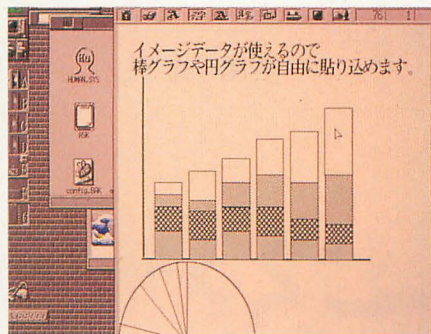
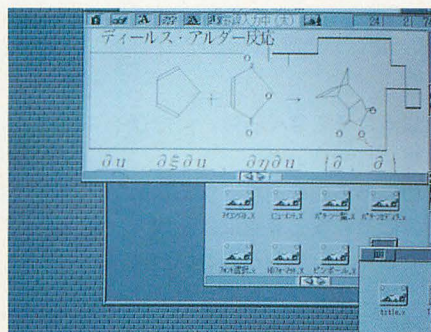
このようにさまざまな画像フォーマットに対応できる

大きさや比率も自由自在だ

動画再生が標準でサポートされた

当然のようにグラフを張り込む

亀の子だろうが，もう怖くない

数式も記述できなくはない。アウトラインフォントも使える

装飾の例。ながーいメニュー

そうに思えるのだが。

新世代の象徴?

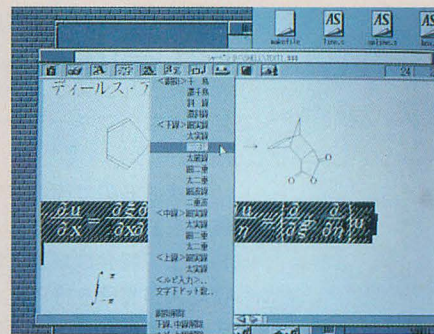
そして新しいSX-WINDOW用テキストエディタ，シャーペン.Xである。従来のエディタ.Xがかなりよくできたツールだったので，「禁則処理さえあれば……」と常々思っていたのだが，ついにそれが実現されたのだ。それだけでもSX-WINDOWの環境が一転するというのに，機能強化はそれに留まらない。

まず，マルチフォントがちゃんとサポートされた。単にいろいろなフォントが交ぜられるというだけの中途半端なものではない。文字間行間が自由に設定可能だからやりようによってはなんでもできる。そして，同時にこれは標準となるリッチテキストフォーマットが現れたことをも意味しているのだ。

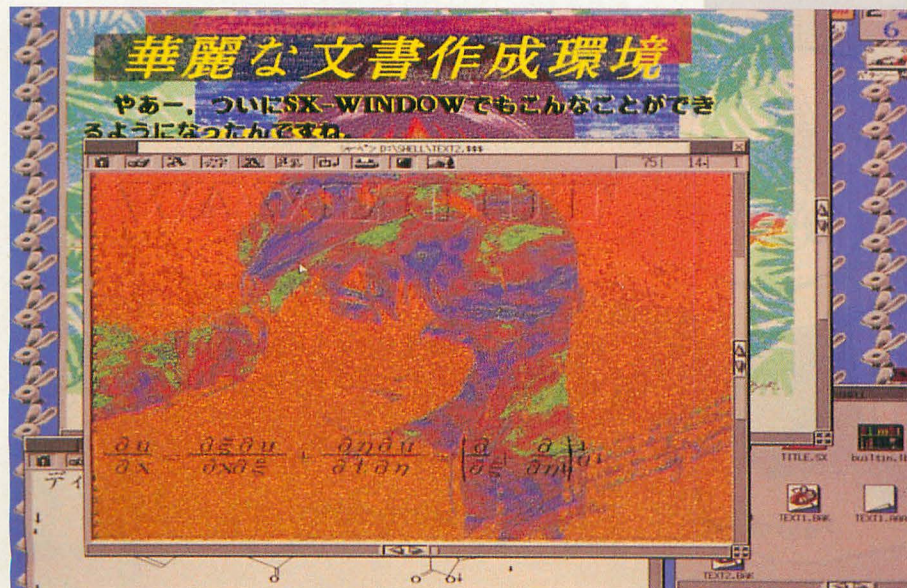
イメージファイルを扱うこともできる。文書内に簡単に図が張り込めるのだ。こうなると，グラフ作成ソフトやドローツールがほしくなる。

オマケでこんなものをつけていいのだろうか?

さらに，キー設定やシステム設定，メニューの内容その他がすべて再定義可能だ。キーバインドをEMACS風にすることもできるだろう。

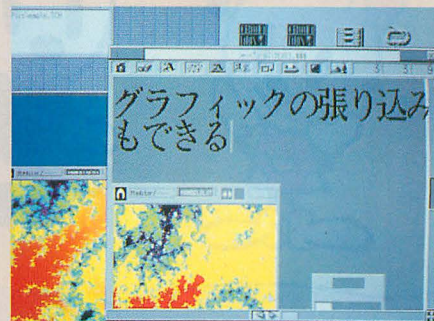
そして外部コマンドによる機能拡張がシャーペン.Xのコアを成していると知り，可能性は無限となった。現状で実現されている機能の多くがアーカイブされた外部コマンドファイルによって実現されているのだ。必要な機能はあとから追加できるのだ。たとえばコンパイラを作った場合，シャーペン.X上からそのままコンパイルするようにできるかもしれない。

シャーペン.Xはきっちりと完成したツールではない。アクセサリを利用することにより，今後どんどん形態を変えながら磨かれていくことになるだろう。その意味では形のない，やわらかいツールといえる。

こうしてシャーペン.Xはエディタであり，ワープロであり，環境となりうる。まさに「早川式繰り出し鉛筆」によるデスクトップ革命が始まろうとしているのか。

Sharp Pencil

グラフィックを張り込んだところ。これはPAT4形式だ

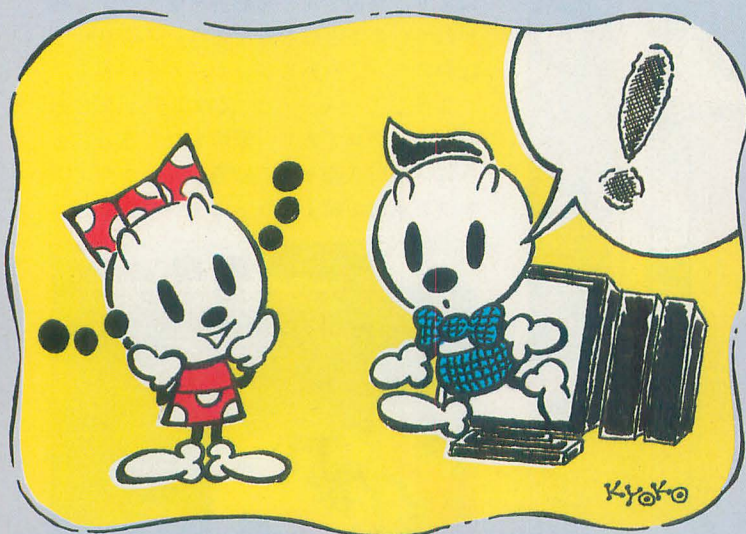
PICを張り込んだ。左が変換前

[第23回]
スターになれるよ

寺尾　響子

響子inCGわ〜るど

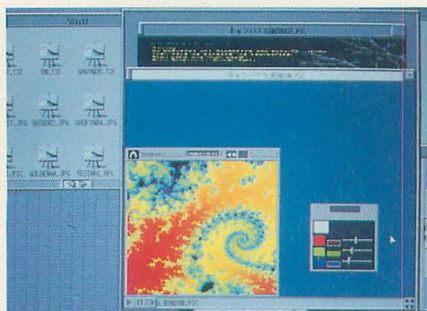
　ぼくらは天才と呼ばれていた。弱冠12,3歳にして，驚異的なプログラミング能力を発揮し，ふつうの大人をはるかにしのぐ収入を得ていたからだ。そのため，かわいくない，ませたガキどもとも呼ばれていたけど。
　ぼくらを仕切っているのは，いつもほほえみを絶やさないマネージャーだ。彼がある日，こういった。
「いまはものが売れない時代だ。我が社も経営の縮小を迫られている。だが，主力事業であるソフトウェア開発は続けていかなくてはならない。そこで人員整理を行い，少数精鋭のプロジェクトチームを再編成することになった」
　にこにこしながら，マネージャーはことばを続けた。だけどぼくは知っている，この笑顔がくせものなのを。
「はっきりいって，君たちの論理的思考力は限界に達している。ふつう，プログラマの能力のピークは30歳ぐらいとされているが，君らにはそれが早く訪れたと我々は判断した。まあ，無理もない。毎日，脳を酷使しつづけてきたからね」

　冷たい話だ。それじゃまるでぼくらは，くたびれたコンピュータみたいじゃないか。
「君らを選別するためにある心理実験をすることにした。音や光を完全に遮断する装置に数時間入ってもらう。その中でじっくり自分と向かい合い，考えたことを報告書にまとめて提出してくれ」
　高級なベンチマークテストってとこだね。ふん。

　そんなわけで，ぼくはここにいる。金属製のシリンダの中の音のない暗闇に包まれて，静かに漂っている。何も聞こえないし，何も見えない。
　できるのは，思いを巡らせることだけ。ぼくは自分の生活についての考察に集中した。まず，記憶の階層を下へ下へと潜る。ときには立ち止まって検証し，不要な項目は忘れる。そして，分類し，統合する。この作業を幾度となく繰り返した結果，だいたいこんなふうであることが判明した。
　学校については義務教育なので，出席日数をクリアしていれば問題ない。成績は体育が少し悪いが，あとはまあまあだ。空いた時間に好きなプログラミングをして金をもらう。父親よりも高額の給料をもらっている以上，家庭に生活費を入れるのは当然だ。そして家庭の機能によって，ぼくの衣食住は保証されている。生活費を差し引いた残りは，母親が手堅い株や債券に替えて資産運用していて，この先数年は収入がなくても家族が十分暮らせるぐらいだ。
　12歳にして，ぼくの生活ルーチンは安定していた。裏を返せば，変化に乏しいということだ。つまり，このシリンダの中と状況はたいして変わらないのである。
　結論。ぼくは仕事をやめる決心をした。ちょうどよい機会だ。もっとも，会社のほうでぼくを不要とするかもしれないけれど。
……何もかも捨てて，もう一度いちから新しい生活を始めたい……
　この１文だけの報告書を提出した。ワープロは使わずに手書きで。ねじくれた汚い字だった。で

も，この字はまぎれもないぼくの字だ。
ぼくは自分の生活を見つけたい。それがどこにあるかはわからないけれど，ビルの自転車は借りずに自分の足で地面を踏みしめて探そうと思う。

数日して，社に出向いてほしいとの連絡があった。あの“笑顔”マネージャーからだった。
「おめでとう。君だけが選ばれたよ。ほかのひとにはやめてもらうことにした。気の毒だが，経営合理化を図るためにはいたしかたない。既存のプログラムの概念をいっさい捨てて，いちからものを作ろうとする頭脳こそ，これからの我が社には必要なのだ。さっそく新しいプロジェクトに加わってくれないか。21世紀に向けて，他社との差別化を図って業界を生き抜くために，いままでとはまったく別次元のコンピュータシステムを開発する予定になっている」
よりによって，やめたいと思っているぼくが選ばれるとは。
「君のためにそれなりの地位と給料を用意した。明日からいつもどおり来てくれるよね。一緒にがんばろう。君なら，低迷するコンピュータ業界を救うスターになれるよ」
マネージャーの声がにこやかに響いた。
ぼくは，ゆっくりとズボンのポケットに手を入れた。報告書とほぼ同じ１文を書いた紙切れを取り出すために。
……何もかも捨てて，もう一度いちから新しい生活を始めたいので，貴社をやめさせていただきます……

SOFTWARE INFORMATION

ここのところおとなしめだったX68000のゲームですが，復活の兆しがチラホラと。「餓狼伝説」や「エトワールプリンセス」といったところの発売も近づいてきましたし，さらに，あそこもそろそろ……。

餓狼伝説

ネオジオ専用格闘アクションゲームとして発売され，第2作まで登場した「餓狼伝説」が，いよいよ4月23日にX68000に登場する。

このゲームは「ストリートファイターII」と同じく対戦型で，1人プレイではコンピュータの操る敵キャラを倒し，勝ち抜いていく。そして，2人プレイでは文字どおり人間どうしの対戦となる。試合は3セットマッチで2勝したほうの勝ち，というのもお馴染みのルール。

内容や操作，登場キャラクターは，ネオジオ版からそっくり移植されるらしいが，Cボタン(投げ)は同時押しで処理される。

X68000版　5″2HD版　8,800円(税別)
ホームデータ　☎078(261)2790

今月から「期待するソフトTOP10」です

1. SX-WINDOW ver3.0
2. 餓狼伝説
3. エトワールプリンセス
4. スターフォース
5. ストリートファイターII & II′
6. Traum
7. 信長の野望・覇王伝
8. 幻影都市
9. シムアント
10. CGAマガジン

はーい。今月から登場の「期待するソフトTOP10」のコーナーでーす。無作為抽出したアンケートハガキから「期待している新作ソフト」欄を集計し，その結果を発表していきます。担当はいままでのTOP10と同じく，私(浦)が務めさせていただきます。どぞ，よろしく。

では，さっそく3月号のアンケートハガキ集計分の結果を発表しましょう。

第1位はX68030とともに姿を現した「SX-WINDOW ver.3.0」。65536色使用可能なカラーや，「シャーペン.X」に魅力を感じている人が多いようです。それから「ASK68K ver.3」に関する声も結構ありました。期待に違わず賢くなったと，スタッフの間でも評判ですよ。

そして，2位。「餓狼伝説」がなかなか多くの期待を集めているようです。ネオジオ専用のゲームで，アーケードでもスマッシュヒットを飛ばした作品だけに，ゲームセンターでハマった体験をもつ人も多いようですね。

格闘モノといえば，本家本元の「ストリートファイターII」の人気も，カプコンから発売の告知がないにもかかわらず盛り上がっています。これが出なけりゃ今年は買うソフトがない，とまで言い切っている人もいますぞ。カプコンさんはユーザーからの視線が痛い毎日ではないでしょうか。

下位のほうでは，10位の「CGAマガジン」が注目株。ゲームとは違った意味でX68000のよさを体験できるという声が届いています。創刊号はすでに発売中なので，次回作への期待票といえるでしょうか。

ありゃ，もう行数がない。ではみなさん，このコーナーをよろしく。そして，気合の入ったコメントをお願いします。（浦）

エトワールプリンセス

ちょっと発売が延び延びになってしまっている「エトワールプリンセス」。どうやら開発は順調に進んでいるようだ。ようやく，ゲーム内容が見える，全12ステージのボスと6ステージまでの敵キャラクターを配置したバージョンが届いたので紹介しよう。

広いマップをちょこまかと走り回りながら敵をばしばし倒していって，最後にボスとごたいめ〜ん。アクションRPGの王道をいくゲーム構成だが，遊んでいるこっちまで楽しくなるくらい，テンポよくゲームを進めることができ，ゲーム展開が非常に快適なものとなっている。ボスを倒すと封印された他部族のキャラクターを仲間にできたり，多少の謎をからめてあったりとRPGの要素もあるが，敵との戦闘や石渡りなどアクションに重点を置いたゲーム作りが基本のようだ。

マップのバリエーションも豊富だし，視覚的演出効果もバッチリ決まっているので見応えも十分。特にそれぞれのステージのボスは，かなりのパターンがありプレイヤーを飽きさせない。

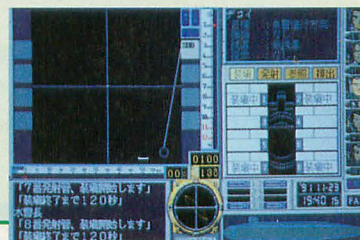
キャラクターはかわいい，ノリは軽い，文字は丸い，でも，落ちるところは落ちる。総合的に見てやさしすぎず難しすぎず，誰にでも楽しめる遊びやすいゲームに仕上がりそうなので，発売を期待していよう。

X68000用　5″2HD版　9,800円(税別)
エグザクト　☎025(247)9160

沈黙の艦隊

で，ようやくX68000版の写真がお見せできる運びとなった。発売はもうそろそろという感じで，現在最終段階の修正に入っている模様。もうしばらく待ってみよう。

おさらいしておくと，このゲームは同名コミックを原作にしたシミュレーションゲームで，独立国家宣言をした原子力潜水艦「やまと」と艦長の海江田四郎が各国を相手に活躍する。

X68000用　3.5/5″2HD版　12,800円(税別)
ジー・エー・エム　☎03(3736)6879

ヴェルスナーグ戦乱

うーん，なかなか出ないなあ。が，新しいサンプルが届いたので，紹介しておこう。このゲームは基本的にはごく普通のRPG。戦闘時には魔法などがグラフィックで表示され，迷宮の中でなにか起こるとパーティの仲間がしゃべってくれたりする。画面写真を見ていただくとわかるとおり，町では左上のマップエリアに情景のグラフィックが表示される。ね，普通でしょ。

X68000用　3.5/5″2HD版7枚組　9,800円(税別)
ファミリーソフト　☎03(3924)5727

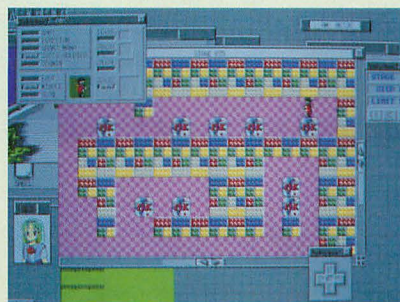
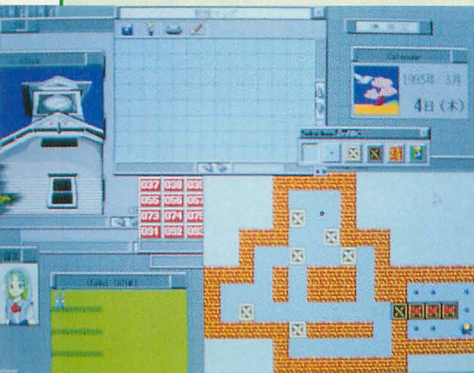
倉庫番リベンジ SX-68K

「ロードランナー」や「フラッピー」などとともに，まさにパソコンの歴史の1ページを飾ったといえるのが，「倉庫番」だ。

で，この名作パズルゲームがまたまたX68000上に甦る。しかも，SX-WINDOW対応だ。常にデスクトップ上に置いておいて，気が向いたら遊ぶというのにピッタリのゲームだから，これはうれしい。

用意された面は，10年間にわたるユーザーからの投稿面など，レベルの高いものが306面。まさに「倉庫番」の最強版といえそうだ。

キャラクターを好きなものに替えることもできるし，操作のアシストや面のエディットといった機能もきっちりと搭載されている。

X68000用　3.5/5″2HD版　価格未定
シャープ　☎03(3260)1161，06(621)1221

●スターフォース

熱き連射ゲーの魂が甦る

Yokouchi Takeshi
横内 威至

アーケードゲームメーカー，テクモ。そのテクモがまだテーカンという名だった頃，「スターフォース」は生まれた。ファミコンにも移植され，人気を呼んだあの連射重視シューティングゲームが，いまX68000で甦る。

ひと昔前のアーケードゲーム「スターフォース」を知っているだろうか。1984年といえばもう10年近くも前なので，知らない人も多いに違いない。その頃は私も小学生だったが，このゲームは名作として記憶に残ったままである。

アーケードゲームはこの頃，シューティング，アクションなどというふうに，ジャンル分けが始まりつつあった。そんななかでこのゲームは，連射モノシューティングのジャンルで最先端に位置づけられていたと思う。失神するほど連射する。腕は痙攣し，体中の筋肉は硬直する。そんなナチュラルハイにひたれるドラッグゲームの始まりは，このゲームに違いない。

撃って撃ちまくって死ぬ

操作は1レバー1ボタンだ。ただ，もう撃つだけだ。かぎりなく現れる敵と地上物を，永遠に攻撃するだけだ。最終面なんてものもない。パワーアップシステムもあまり存在しない時代のゲームなので，おまけ程度のパワーアップがあるだけ。あるのは，合体するとショットスピード，移動の前後スピードが上がるアイテムのようなものぐらい。そのときにBGMが変わるのが，ちょっとうれしい。

ステージの概念はあるが，撃ち落とした数が一定数に達するとステージが終わる。確か地上物50個だったかな，これだけ壊すと単純な要塞（ボス）が現れる。右や左に動くだけのヤツだ。そいつを倒せば，ネクストステージ。ボスの動きは右か左だけど法則がある。でも教えない。自分で見つけてもらいたい。

各ステージの画面は渋い。アルファから始まりベータ，ガンマ，伝々と続く。となると，最後がありそうだが，すべて越えるとインフィニティなのだ。このときの感動は並々ならぬものだが報われない。涙しつつも永遠に突き進むのだ。

では何が目的かって，それは失神するのが目的だ。このセンは譲れない。決して連射スティックなんて使ってはいけない。とはいっても，まだまだ初期の“連射ゲー”だから，そんなにはハードではない。ちょくちょく固い敵が並べてあったりするので，ここぞというときにそいつらを粉砕すべく，ゴッドハンドを使用するのがベスト。きっといつかは酸欠に陥り，かぎりない疲労とエクスタシーが得られるはずだ。

また，この連射により，実にナイスなボーナスが用意されている。

たまに，合体キャラが出てくる。音楽が変わるのでそれが合図だ。しかし，あわてて撃ってはいけない。ひと息つき，痙攣撃ちの用意をするのだ。しばらくすると，画面中央上部にコアが現れ，四方からパーツが集まる。中心が輝いたら一気に撃ち込む。合体前に8発，秒間16連射のなんとか名人なら，コンマ5（0.5）秒弾を食らわせれば50,000点。失敗して，合体してしまうと壊しても1,000点。こいつは泣けるぜ。フライングもペナルティだ。輝く前に撃ち込んだ分だけ，余計に撃ち込まねばならなくなる。痙攣マニアなら，ちょっとぐらいハンディをつけてやるのもいい。

緑に輝く地上物は4発で1,000点。だいたい連なって置いてあるから燃える。そして，中盤のエリアにこいつが横2つずつ，縦に長々と並ぶエリアがある。こいつは最大のエクスタシーポイントだ。どちらかの列を連続で壊すと，80,000点のボーナスだ。両方なら，10個連続ぐらいで150,000点っていうのがあったと思うのだが，技術不足で確かめられなかった。無念。ここは敵の攻撃も容赦ないから，運もともなわねばボーナス獲得は難しい。

しかし，うまくステージクリアに絡めてやると少しは楽になる。前述のとおり，一定数の敵を破壊すると要塞出現だが，その直前には音楽が変わるとともに敵が出なくなり，しばらくスクロールしているスキができる。これを利用する。がんばれ。

そして，もうひとつ。誰もがアツくなったのが“？”プレートなのである。これは8個ほど横に並んでおり，そのうちのひとつに1UPが隠されている。やはり4発で破壊，ではなく裏返ってくれる。ニコニコマークとでもいうのか，そんなヤツが出ると自機が1機増える。ラッキーですな。

これも，ある法則により1UPとなるプレートが決められる。スコアに関係する，とまではいっておくが，タイミングよくスコアをチェックするのはきつい。実は私もすでに覚えていないから（無責任モード），各自がんばってほしい。例によって，敵の猛攻の真っ最中に現れる。いやらしい。たとえうまく出てくれても，撃ちもらした敵に逆襲されるパターンが多い。ナインボールで素晴らしいショットを見せつけたものの，

X68000用　5″2HD版　4,800円（税別）
電波新聞社　☎03(3445)6111

地上物でさえカタイものが多い

9を落としたと同時に手玉まで落としてしまうケースに似ている。

ということで，このゲームにおいて目指すことは3つだ。まずはステージインフィニティ，そしてハイスコアだ。現代のゲームと違って，スコアはプレイ技術と密接な関係をもっている。カンストプレイヤーもいるらしいので，とにかくハイスコアだ。だが，目指すはなにより痙攣だ。男なら血管切れるまで連射しろ。女もだ。

合体キャラ。中心部が赤く光ったら撃て

左の小さいのがアイテム

連射の鬼となり死ね

いつの間に自動連射というのが定着してしまったのだろう。かつて連射は指，および腕を鍛え上げるためのスポーツとして広まっていたのをご存じであろうか。連射ゲームの発祥はおそらくコナミの「ハイパーオリンピック」。この「スターフォース」よりさらに古く，過去最大の“連射ゲー”として知られる。3ボタンしかないコンパネは，まさにボタンを刻むこと以外は許されない，鬼のゲームであった。2つのランボタンを連射することにより，競技者がスピードアップするのは明白。しかし，ボタンを押せば弾が出るというほどの視覚的な反応はないので，己の体力，精神力がモノをいう高貴なゲームなのである。

初期の連射は痙攣ではなく，パワーで押すのが正しかった。代表的な連射方法は，ランボタンを左右の手でタイミングよく叩きつける方法。その絶妙なリズム，圧倒されるパワーはツーバスのドラムを超越する。このタイプの人間は現在ドラマーを職としている可能性が大きい。以後，この連射が最も一般的になり，1ボタンゲームの痙攣法に応用されていく。腕に力をこめ，微妙な強弱をつけて指を震わせる。これは伝統的な連射法として，今日まで伝わる。

また，そのほかにも特有の連射があみだされていたのも懐かしいかぎりだ。応用されず，歴史に残るのみとなった方法に，振動粒子速射法が挙げられる。必殺機器，鉄定規をボタンの先に当て，端をタイミングよく弾くと，それは素晴らしい連射へつながった。だが，ジャンプボタンをサポートする人員が必要なうえに，器具を使うのは邪道とされて，闇へ葬られてしまった。

そして，もうひとつは“こすり”だ。指を立てて，ひたすらボタンをこする。ボタンのダメージが大きく，お店では嫌われがちな方法だが，エキスパートでなくても確実なスピードが得られるため，愛用されているようだ。このこすりを極めた人間はもちろんラップに目覚め，各地でDJの職を勤めていることはあまりにも有名。というのはまあ冗談だが，連射が常に時代をリードしつづけたことは，皆理解できたはずだ。

ボスは面が進んでもあんまり変わらない

Hにメロメロで死ぬ

出た当時に，発売元のテーカン（現テクモ）は簡易攻略マニュアルを配っていた。しかし，そこには全国のゲーマーをよりクレイジーにする謎が書かれていたのだ。これこそ，このゲームの最大の超H，隠れキャラクターの存在だ。書かれていた内容は，エリアラストにあるといわれる謎の象形文字，そしてゴーデスと呼ばれるエジプト顔であった。全マップ中のどこかにこいつが隠されており，100万点ボーナスに関わるキャラクターだということで，ゲーマーは先を争って探したものである。ばらすとあまり面白くなくなるが，古くからのゲーマーの間では常識であるから，ヒントだけ教えよう。つまり，“生きた化石”のそばだ。どうせゴーデスは簡単には壊せないから，知ってるほうが熱いはず。また，ここのエリアで死ぬと，再びループしてきて戻るまでチャンスはない。かぎりなくシビアであるこのボーナスをつかんだとき，初めてこのゲームを極めたといえよう。

いま思うと，これ以前から隠れキャラってのはあった。古くは「ゼビウス」，そして「ドルアーガの塔」なんてのも。でも，やりすぎで“隠れだけのゲーム”が出てきたためにすたれてしまった。

ナイス移植に驚喜して死ね

基板から読み出したのかどうだか知らないが，敵のアルゴリズムだとかタイミングだとかは，もうまったく完璧でしょうな。もちろん，いま流行の384×256モードでアーケードモード対応だし，ディップスイッチと同じ設定ができるし，まさに完全移植。シンプルさ，細かくないアバウトなデザインなど，懐かしさ以上に新鮮ささえも漂うこのデキには，もう驚喜するしかないです。名作と呼ばれるゲームを完璧に遊べるのはX68000だけ。電波さんはもう神様です。

懐かしすぎて死んでしまう

いやー，第2弾には驚かされたけど，第3弾もなかなかナイス。久しぶりに腕を震わせて，アツくなってしまった。今後も続くこのシリーズに期待してしまう。

同時期にはパワーアップの元祖，「B-WINGS」(DECO)なんて傑作もあるし，1985年あたりのジャンル分けが始まる以前のゲームはシブいと思う。これ以降，あまり面白いモノがなくなってしまった，と思うのは気のせいか，それとも年だろうか。

1985年以前のゲームを思い出すと，ナムコなら「グロブダー」「フォゾン」「リブルラブル」なんてアイデアの秀逸なものもある。コナミはまだマイナーだったけど，「スクランブル」などに「グラディウス」の原点が見いだせる。ああ，あの頃はよかったよな。

個人的な希望は以上述べたとおりのものに加え，SNKの出世作「TANK」「怒」，そして関係ないが「トリオザパンチ」。さらにもうひとつ，なんといってもリンチゲーム「ピットファイター」である。このシリーズのさらなる発展を期待したい。

総合評価 (0 / 5 / 10)

項目	評価
ネームバリュー	★★★★★★
移植度	★★★★★★★★★★
連射のアツさ	★★★★★★★★★
シリーズへの期待	★★★★★★★★★★

モテモテは帝国建設の礎です

Urakawa Hiroyuki
浦川 博之

モンゴルの高原を駆け回り，蒼き狼と呼ばれた，チンギス・ハーン。この「元朝秘史」では，かの英雄の生涯をたどりながら，世界制覇を目指す。勇敢さはもちろん大事だが，人を愛することも同じくらい重要視せねばならない。

「蒼き狼と白き牝鹿」といえば，「信長の野望」や「三國志」と並んで，光栄の歴史シミュレーション3部作に数えられるシリーズである。が，いっちゃなんだけど，いままでは影が薄かった。

しかし，光栄はその逆境にめげず，この「蒼き狼と白き牝鹿・元朝秘史」をリリースした。ゲーム内容には相当の自信があるに違いない。ちなみに，私のモンゴル予備知識は，焼肉のジンギスカンとモンゴリアンチョップだけ。あ，あと「ジン，ジン，ジンギスカン」ていう歌。はたして，こんな私でまともに遊べるのだろうか？

やっぱりアレですか？

「元朝秘史」も光栄の歴史シミュレーションのひとつだから，基本的な部分は「信長の野望」や「三國志」などと一緒だと考えていい。違うところは大きくいって，次の2つだけだ。

ひとつはマップが2種類あること。片方はモンゴル高原を舞台にしたもので，モンゴル統一のシナリオ用。もう片方がユーラシア大陸全体を舞台にした，世界制覇編だ。さらにこのゲームには「ユーザーシナリオ」というシステムがあって，モンゴル高原を統一するとその時点でのデータをもってユーラシア大陸のマップに移動できる。有能な武将をたまたま見つけたとか，優秀な子供がたくさんいるというときにはメリットが大きい。それにチンギス・ハーン以外の武将でプレイしても，世界制覇にチャレンジできるのだ。プレイのバリエーションが広がるのはいいことである。

もうひとつが，知らない人でも知ってる「オルド」。戦争して占領した国にお姫様がいた場合は，オルドに入れてお妃にできるというアレですな。そして口説くと，1年後に子供が生まれる（こともある）。もう，さすが光栄。「団地妻の誘惑」はダテじゃない(古い)。たしかアダルトソフトでもこんなシステムがあったような気がするが，そこは光栄，ただエッチなだけじゃないぞ。オルドはゲームシステムの一部としてちゃんと成立していて，お妃を口説けないとマジに国の将来が危ないようになっている。オルドをバカにするのも期待するのも勝手だけど，どちらにしてもプレイを始めたら結局は「うおおーっ，子作りじゃー！」となるのはさけられないのよ。

実践プレイ始めます

シナリオ1の「モンゴル高原の統一」をプレイしてみよう。武将は4人の中から選べるが，私はテムジン（チンギス・ハーンの幼少の名）以外はどんな人かよく知らないので，テムジンで始めてみる。

まずは国を豊かにすることからだ。この「元朝秘史」はほかの歴史シミュレーションに比べて，軽い作りになっている。国を豊かにするといっても，することは豊かにしたい分野に住民を割り当て，施しをしてモラルを高くするくらい。コマンドも少なければ国のパラメータも少ない。それさえ面倒臭い人には，武将を政治顧問として任命しておけば，内政の肩代わりまでしてくれるのだ。もっとも，それじゃゲームクリアはいつになるかわからない。ちなみにテムジンにはボオルチュという頼れる政治顧問がいる。

迷ったときは顧問に相談してもいい

余談だけど，モンゴルの名前って結構独特だ。慣れるまでは部下や姫を区別するのにちょっと苦労する。チンベとチラウンとか，イェスイとイェスゲンとかね。さらに世界制覇編に進むと，変な名前は隆盛を極める。実在しない武将をコンピュータが合成するので，とんでもない名前が次々と登場するぞ。グフヘガイとかヒミルベトラとか，セッセホラージャだって。おまえら，新種のバクテリアかー！

さて，モンゴル族は最初は弱小勢力で，軍隊の部隊数もほかより少ない。ゲームを始めると，すぐ周りの国から貢物や帰順を要求される。黙っているとじきに攻め込ま

部隊どうしの戦いはクオータービュー

X68000用 3.5/5"2HD版3枚組 9,800円(税別)
光栄 ☎045(561)6861

れる。その前にこちらから叩くべし。なにも1回の戦闘で勝つ必要はない。このゲームでは1部隊200人となっているが，1人でも生きて帰れば次のターンにはなぜか200人に戻っている。よって全滅をさけつつ相手の部隊をつぶせば，いつかは勝てる。たとえそれができなくても，敵国の城や街で補給を行えば，金や食料を奪ってくることができる。そうとわかったら，あとは“いざ戦争！”だ。

大きな口の叩きあい。試合前のボクサーみたい

オルド。本妻のボルテはわりにすんなりと

必殺異国格闘技戦

「元朝秘史」は，戦闘もライトな作りである。火計も籠城もない。「ない」といってもけなしているわけじゃなく，私はむしろこのオタッキーでないシステムを気に入っている。勝利条件は敵の拠点をすべて占領するか，相手を全滅させることだ。

もっとも，戦闘シーンはあまりゲーム性があるとはいえない。コンピュータの頭があまりよくないせいで，攻め込んだ人間側が圧倒的に有利なのである。詳しい話は省くが，要するに籠城している敵に対して，蒙古騎兵のヒットアンドアウェイ戦法が確実に決まってしまう。いったんこれがわかってしまうと「勝つか負けるか」という部分が消えて，戦闘がただの作業になってしまうのだ。モンゴル高原編では，弓をいっぱい持った蒙古騎兵が3部隊いれば，8部隊ぐらいの敵兵に立ち向かうことができる。もっとも世界制覇編になると，部隊数が増えて戦況が複雑になり，そうもいかなくなるのでまあ許そう。

モンゴル高原では部隊の種類は騎兵と歩兵しかないが，世界制覇編になると火砲兵とか象兵とか，武士に騎士とバラエティに富んだ部隊が戦いを繰り広げてくれるのだ。火砲兵が日本の武士を苦しめたりすると，歴史マニアでない私でも，ちょっとワクワクする。

テムジンはオルドにおるど

作りがライトだとはいったが，逆にやりたいことができないともいえる。たとえば領土が増えても，自分が治められるのは1国だけなのだ。ほかの国は領主を任命してそいつに任せるしかないのだが，この領主というのがよく裏切るのである。ちなみに忠誠度はこのゲームにはない。どの部下も等しく裏切りの可能性を抱えている。

そのなかで絶対に自分を裏切らないのが親族。自分の息子や娘の婿たちだ。つまり，国家の安定を図るためには自分の子供がたくさん必要になる。ここにオルドの必要性が出てくる。すべては国のためなのだ。

「そなたの瞳は満天の夜空の星のようだ」

「私が夜空の星であったら，毎夜毎夜あなたにお会いできますのに」

……国のためなのにぃ。

オルドではお世辞をいったり自分を売り込んだり，愛を語ったり贈り物をしたり，あらゆる手を使ってお妃を口説こう。気に入られるとゲージが増えて，MAXに達するとムニャムニャできましたというわけ。

子供には母親と父親の形質が受け継がれるそうなので，なるべくなら優秀な女性のもとに通いたい。が，だれが優秀かは子供が生まれてみるまでわからない。ゲームを何回かやって，経験で導き出すしかない。とりあえず，本妻のボルテはかなり優秀な子供を生んでくれる。求めにもすぐに応じてくれるし。

しかし，これですみゃあ問題はないんだが，特に世界制覇編になったりするといろんな文化圏の女性がやってくるので，ついついグラフィックが見たい（別に過激なわけじゃないよ）ために，あるいはお気に入りができたためにオルド通いが始まってしまったりする。見さかいなく口説き始めるともう政治のコマンドが出せない。「国家は女で滅ぶ」を地でいっちゃったりする。

ちゃんとちゃんとの元朝秘史

やや説明の足りないところもあったけど，シミュレーションで進め方を説明しているとほとんどルール解説になってしまうのでこのへんで勘弁してちょうだい。ほかの部分は，いままでの歴史シミュレーションとほぼ同じだと思ってもらえればいい。ただ進歩したなと思うのは，ゲームの進行につれて，プレイヤーと同様に勢力を伸ばす陣営が出てくるところだ。たいていのゲームでは後半はほとんど弱小国つぶしになってしまい，勝ちが見えてヤル気を失うことが多かったが，このゲームはそういった“中ダルミ”が起きにくくなっている。メニューの出し方が変わったり，兵隊に関するデータテーブルが参照できたりと，前に光栄が発売したゲームよりも確実に進歩している。マニアックに走らず，幅広い文化圏についてしっかりと練り上げた光栄ワールドに，ぜひとも足を踏み入れてほしい。

ゲームの完成度は一級品

題材がマイナーな分，思い入れがなくても楽しめるように，受け口の広いゲームデザインがされている。何も知らない私でも十分楽しめた。ゲームデザインとしてはほぼ完成の域に達している。自動化がだいぶ進んで，わからないときはとりあえずコンピュータにやらせてみるという遊び方が可能になっている。

効果音がPCMになっていて，再生のためにゲームの進行が止まるとか，あるいは金子氏がいつも指摘しているディスク交換時の配慮の問題とか，プログラム周りはまだまだ鍛える余地がありそうだ。最終的には画面の切り替えや，数字の表示がなくてもビジュアルで理解できるところまでインタフェイスを詰めてほしい。とりあえず，半透明のマウスカーソルは見づらいので不可。

総合評価	0 — 5 — 10
操作性	★★★★★★
グラフィック	★★★★★★★★
オルドの楽しさ	★★★★★★★★★
音楽	★★★★★★
国際性	★★★★★★★★★
熱中度	★★★★★★★

アリと遊ぼう!

3月号に引き続き，「シムアント」のレビューをお送りする。前回はクイックゲームを紹介したので，今回はフルゲームと実験モードをほじくってみよう。いろんな種類のアリになりながら，アリの生活を満喫しよう。

Nishikawa Zenji
西川 善司

アリは身近な昆虫だが，その生態を深く知っている人はごくまれである。アリはその見かけ以上に高等な昆虫なのだ。人間社会における人間のように，各アリはアリ社会を存続させるためのひとつの歯車として機能している。領土拡張や裕福な土地を獲得するために侵略戦争をしかけることもある。ただ人間と決定的に違うところは，そのすべての行動が「冷酷」になんのためらいもなく行われることだ。

巣を警護するアリは，近づく外敵を文字どおり命を賭けて噛み殺そうとする。それがたとえ，自分より大きく勝ち目のない相手であってもだ。我が身よりも，自分たちの「社会存続」を第一に考えているのだ。

恨みつらみを思うことはないが，喜びや幸せを感じることもない。「シムアント」は単なる“ありんこさんのげーむ”ではなく，“冷酷機能重視社会存続シミュレーション”という見方が正しいと思うのは私だけだろうか（待つ！突っ込み）。

フルゲーム序盤攻略

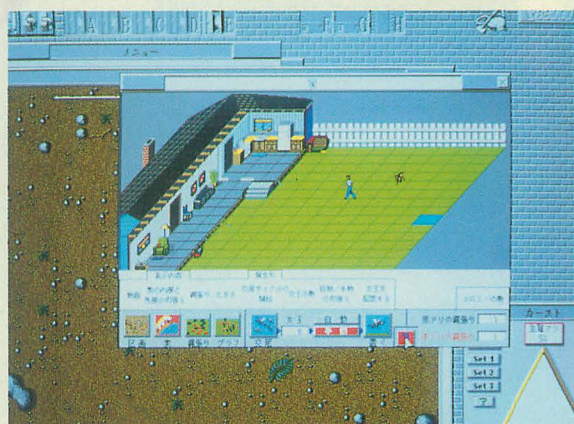
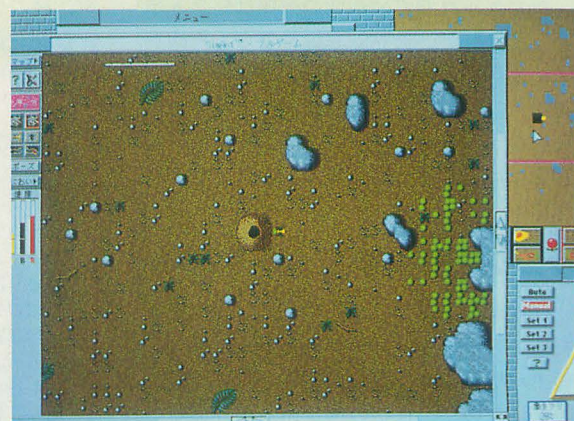
アリが大量に発生して人間の家を襲う。そんな悪夢をアリの立場から描いたゲームが，この「シムアント」のフルゲームモードだ。人間が住む一軒家（写真参照）をすべて黒アリの巣で埋めつくし，宿敵の赤アリ，そしてこの家の主，人間までも追い出すことがゲームの目的だ。前回紹介したクイックゲームは，この家の庭のほんの一区画での戦いにすぎなかったのだ。

クイックゲームでは，女王アリと，自分が操作する働きアリの2匹でゲームが始まるが，フルゲームではまずプレイヤーは羽アリを操作することになる。ゲームフィールドのどの位置に巣を作るかを指定しなければならない。ゲームが開始してからすぐに巣を作ってもいいが，マップでゲームフィールドを確認し，食料(マップ中の緑のボール)に近いところまで移動してから巣を作り始めるのがいいだろう。くれぐれも移動中に死なないように。

巣を掘り下げたあとでメニューから繁殖開始を選ぶと，羽アリが女王アリになり最初の卵を産む。すると，それがすぐ孵化して「クイックゲーム」でおなじみの黄色いプレイヤーアリが生まれる。さあ，食物をフィールドから採集して巣へ運ぶことを繰り返し，ほかの仲間アリの行動を軌道に乗せよう。前回もいったように，プレイヤーアリが巣から食物への道を何度か往復すると，ほかの仲間アリもその行動を真似するようになる。ゲーム序盤の展開の善し悪しはプレイヤーアリの行動にかかっているのだ。

フルゲームはクイックゲームとは違って，最初のマップで登場する赤アリを撲滅してもゲームクリアとはならない。フルゲームでは，まず「人間を領土から追い出す」ことが第一の目的で，これを果たさないで赤アリを撲滅しても，ゲームの都合上か，赤アリは庭の別のエリア上にゾンビのように蘇ってしまう。人間を追い出したあとでないと，赤アリの排除はあまり意味がないようだ。

というわけで，序盤ではクイックゲームと違って，それほど躍起になって赤アリと争う必要はない。まずは，安定した巣の運営が目的だ。それにはとにかく労働力と食料の確保が先決だ。

仲間が増えないようであったら，△B (Behavior Control)の行動コントロールの「看護」の割合を多くしてみよう。また，マニュアルにもあるとおり，兵隊アリや生殖アリは働きアリよりも多くの食料を消費する。よって序盤は巣がそこそこ賑やかになるまで△C (Caste Control)のカーストコントロールは働きアリの出産率をかなり

ひひひ，この家を占領するのだ。犬もいる

エサを運んで，東へ西へ

X68000用 5" 2HD版　12,800円(税別)
イマジニア　☎03(3343)8911

多めに設定しよう。極端な話，働きアリが100%でもいいだろう。

余裕ができるまでクモや毛虫，アリジゴクなどに対しての攻撃はさけよう。毛虫は確かに無抵抗で格好の獲物だが，仕留めるまで時間がかかるのであまりオススメできない。それならば，ちょっと歩いていって食物を拾ってきたほうが効率がいい。クモやアリジゴクを撃退するには多大な犠牲を必要とする。あまり，無謀な戦いは挑まないほうが身のためだ。

実験モードでは壁を使って迷路も作れる

さらに，アリの増減も可能

女王アリの旅立ち

フルゲームでは新しい女王アリを誕生させて，これをほかのエリアへ送り込み，種族の領土を広げることができる。自分の巣の景気が潤ってきているようであれば，さっそく新女王アリを作ってみよう。

女王アリは羽アリのメスが羽アリのオスと交尾することにより変態したものだ。だから，女王アリを作るにはまず生殖アリを出産しなければならない。そういうわけで生殖アリの出産率を上げてみよう。やがて，画面に羽のついたアリが目につくようになったなら，庭マップを出して“交尾サイクル”のアイコンをクリックしよう。しばらくすると，交尾によってメスの羽アリが女王アリに変態する。しかし，巣が食料不足だったり看護が手薄だったりすると，なかなか新女王アリは誕生しない。しばらく待っても女王アリができないようであれば，あきらめて巣の繁栄に努力しよう。

女王アリが誕生したならば，「手動で置く」をクリックし，女王アリを庭の好きな区画へ送り込もう。好きな位置とはいっても，女王アリは体重が重くあまり遠くへ飛べないため，そんなに遠くに送り込むことはできない。すでに赤アリの領土となっているエリアへも送り込むことはできるが，それはゲーム終盤になるまでやめたほうがいい。なにしろまずは人間を家から追い出すことが先決なので，家の中へ向かって敵のいないエリアを1つひとつ確保していき，確実に領土を広げていこう。

人間の家を70%以上占領すると，人間を追い出したことになる。これを達成したら，あとは赤アリの撲滅がゲームの目的。今度は赤アリの領土に対して自分たちの新女王アリを送り込む必要も出てくる。全体マップ上では赤アリが支配している領土は赤，プレイヤー側(黒アリ)が支配している領土は黒，また，両種族が対立している区画は茶色で表される。プレイヤーはこういった前線区域に乗り出していき，指揮をとる必要がある。人間を追い出したあとのフルゲーム後半は，クイックゲームのハードランク相当がたくさんという感じだ。実はこのフルゲーム，人間を追い出してからの道のりが長いのであった。

実験モード

実験モードは，フィールド画面上でアリを好き放題イジメることができるモードだ。写真にあるようなメニューを使って，好き勝手にシムアントの世界を編集することができる。壁を使って迷路を作り，アリをそこへ放り込んでみたり，アリジゴクに突き落としてみたり，赤アリの巣に黒アリをばらまいたり(もう大騒ぎ！)，アリの巣の出入り口に殺虫剤を撒き散らしたり，もうなんでもアリの世界。

適当にいじめたあとで実験モードを終了し，その世界をクイックゲームとしてプレイすることもできる。クイックゲームやフルゲームでのプレイのコツがつかめない人は，このモードで適当にゲームバランスを調整してからプレイするのもいいかもしれない。たとえば，赤アリの巣に石をばらまいておくとか殺虫剤を撒いておくとか，自分たちの巣の近くにエサを撒いておくとか，黒アリの味方をあらかじめ作っておくとか，工夫次第でいかようにでもゲームバランスを調整できるだろう。

まとめ

シムアントは「マップ・ウィンドウ(全体地図)」と「クローズアップ・ウィンドウ(ゲーム画面)」や「行動コントロール・ウィンドウ」「カーストコントロール・ウィンドウ」など，いくつものウィンドウを開いてプレイしなければならない。PC-9801版などではいちいちウィンドウを呼び出さなければならないのだが，X68000版ではSX-WINDOW対応なので各ゲームウィンドウを広大な実画面の任意の場所にばらまいておくことができる。必要に応じてスクロールさせて使用することができるので，その点ではなかなか快適といえる。

前回，X68000版のシムアントは遅いと書いたが、あれはサンプル版だけのことで、製品版は格段に速くなっていた。X68000 XVIならまったくストレスを感じずに遊べるし，10MHzマシンでもいろいろと無理をしなければ，なかなか快適に遊べる。さすがイマジニアだ。まだまだ10MHzマシンもがんばるかな。でも初代機ユーザーの私は，そろそろX68030がほしいわーん。

シムアントをいっそう楽しく

このゲームには，アリたちが人間の言葉をしゃべるおしゃべりモードがある。「ミツバチマーヤ」とか「みなしごハッチ」のようなメルヘンチックなものではなく，結構ブラックなセリフを各アリが吐くのは笑える。たとえば黒アリが赤アリとご対面すると，「お前たちの女王はシロアリと寝る！」「赤アリになるくらいなら死んだほうがましだ！」などとお互いを罵るのだ。見ていてなかなか滑稽である。また，このモードではクモも「おいしいおいしい！　アリの内臓！」とかいってしゃべる。こういったユニークなアイデアとセンスは“さすが西洋人！”いやん，ばかーんという感じだ。

ところで，この「シムアント」はなかなか奥が深い。前回紹介した警告フェロモンの意外な使い方をはじめ，クモの誘導テクなど，まだまだ未知なるテクニックが隠されていると思われる。何かわかったらレポートしてほしい。あと，マニュアルにも紹介されている「ミステリーボタン」アイコンって，いったい何だろう？

レンジでチンする格闘技

Shiba Mamoru
司馬 護

「ストライクレンジ」，それはいくつもの階層に分かれた高層スタジアムで行われる，ロボットどうしの戦闘競技。上に昇ったり下に降りたりしながら，敵の動きをうかがい，照準に捉える。狙いを定められる前に，敵を倒せ。

西暦2020年，競技用の機動兵器によって行われる疑似戦闘競技「ストライクレンジ」に，地上の人々は熱狂していたんだと。5年後の日本ですら満足に想像できないテレビ番組があるなか，27年後の予言をするとはなんて大胆な。5年後はフィクションになるけど，27年後だとSFになるのはなぜだろう。人類の夢なのかなあ。

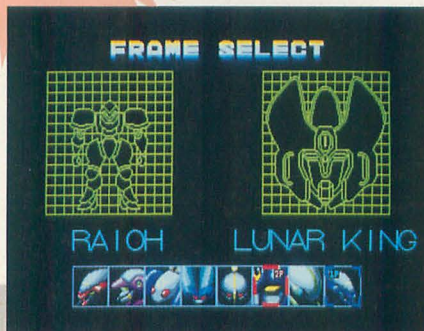
「ロボットは8体」「2人対戦モード付き」「白熱興奮のバトル」。いやがうえにも思い出してしまう，あのゲーム。

ダメージ制だし，タイム制限もある。同キャラ対決もできればキャラごとの必殺ワザも違う。対戦後に相手の顔が出てくるし，メッセージまである。

しかし，まるで別ものであることは断っておこう。みんな似たり寄ったりのゲームデザインじゃつまらないしね。

ガンダムなんだな，これが

さて，ロボットの操縦方法は非常に簡単だ。基本に忠実で非常によろしい。違和感なくゲームの世界に入ることができるだろう。移動は上下左右。左右はレバーを2回同じ方向に入れるとダッシュになる。

ロボットは人型，戦車型，クモ型，顔型などがある。予想を裏切らず，人型は高機動，戦車型は重火器，クモ型は意外にすばしっこくて，顔型はサイコミュ搭載である。ああ，華麗なるガンダムの世界。

床の上をうまく動き回りながら，敵を撃つ

X68000用 5"2HD版2枚組 4,800円(税込)
ブラザー工業(TAKERU) ☎052(824)2493

選べるロボットは8体。強いのを見極めよう

ちなみにダッシュできるロボットとできないロボットがいる。どうせなら上下にもダッシュできると，画面の中を縦横無尽に動き回れるのにね。

Aボタンは通常攻撃，Bボタンが防御，A＋Bで特殊攻撃になる。通常攻撃は近距離と遠距離で変わるので，トータルで3種類（近/遠/特殊）の武器を搭載していることになる。残念ながら，蹴る，殴るという動作はないようだ（戦国魔人ゴーショーグンじゃないぞ)。あくまでも武器で戦うのが基本らしい。そのせいか武器の種類は豊富で，レーザーやミサイル，バーナー，機雷などがあり，なかにはファンネル(ビット)のようなものまである。ビームサーベルとか，ヒートロッドがあってもよかったのにね。それはちょっと子供だましの気もするけど。

ちょっと○が小さい

具ではない。「画面狭しと動き回るロボット」というと聞こえはいいが，言い換えればキャラが小さい。ショット系専門のゲームなので，近接戦はあまりないのはわかるが，ある程度距離を置くためにキャラが小さくなってしまうのはもったいない。

細かい演出をしたとしても，わかりづらいしね。私は子供だから(?)，合体・変形ものとかが大好きだし，ミサイルなんかはポッドがガチャンと開いてから発射されると，「かっちょえ〜」とか思ってしまう。えてして，こういう小僧のほうがロボットもののファンだろうし，購買層の中心なのではないだろうか。私が気がついた演出は，ホバリングしているロボットの空気が揺らいでいたところくらいである。もっとファンサービスしてほしかった。

対戦やってりゃニコニコ

対戦が用意されているゲームの最大の共通点は，対戦のほうが100倍は熱いということだろう。このストライクレンジでも同様である。

まだサンプル版ということもあるだろうが，シングルプレイでの作り込みが足りない。初めて遊んだときも1コンティニュークリアだったし，要領さえつかんでしまえばクリアは簡単である。

また，ロボットの強さに偏りがあるので，弱っちいロボットだとかなり苦戦を強いられる。ザクとジオングくらいの差はあるぞ。初期の「ドラゴンスピリット」をノーミスでクリアするような，ニュータイプレベルの反射神経をもった人でも量産型のザクでは勝てないだろう。

玄関開けたら2分で○○○

狙いは面白い。本文中でも書いたが，サンプル版のプレイということで，製品版は改良されていることもありうる。現段階では若干のバグもあるようだし。音楽は何曲か用意されているようだけど，特徴がないのでイマイチか。MIDIにも対応していないようである。最近，"Oh!X LIVE in"への投稿のレベルが高いので，単に耳が肥えているだけかもしれないけどね。「頑固な男はきらいですか？」。

総合評価	0 — 5 — 10
システム	★★★★★★
スピード感	★★★★★★★
グラフィック	★★★★★★
音楽	★★★★
ガンダム度	★★★★★★★★

TREND ANALYSIS

今月はページの都合で1ページだけとなってしまったが，とりあえず集計結果はお知らせする。右下の表は，アンケートハガキを1000枚無作為抽出して,「最近買って気に入ったソフトは？」の欄を集計した結果である。

最近はソフトを買っていないという人が39%で，他機種のゲームを記入していた人も5%いた。さらに「CGAマガジン創刊号」「X68000 Develop」「電脳倶楽部」を挙げている人が合計で6.5%に及んだが，これはベスト10から外させていただいた。

全体としては，前回までの店頭集計と大幅に変わることはないようだ。新作として「チェルノブ」と「DALK」が入っているが，それ以外は変化なしとしてしまっていいだろう。

店頭集計と比べると，この集計方法では1人あたり1本しか投票(?)できない，ということになるが，その分，純粋な意味での人気度が得られる。

とりあえずはこの集計を続けていきたいと思うので，吟味して記入していただきたい。まあ，「そんなにソフトが買えない」といわれると，つらいところだが。

1993年3月号のハガキ集計ベスト10
最近買って気に入ったソフトは?

POINT	タイトル	発売元	発売日
147	オーバーテイク	ズーム	'92/11/20
44	ストライダー飛竜	カプコン	'92/11/27
32	チェルノブ	電波新聞社	'93/1/29
21	MATIER	サンワード	'92/10/9
18	テラクレスタ/ムーンクレスタ	電波新聞社	'92/11/20
17	グラディウスII	コナミ	'92/2/7
15	DALK	アリスソフト	'93/2/10
14	ロードス島戦記II	ハミングバード	'92/11/20
12	キングス・ダンジョン	ソフトプラン	'92/11/25
11	ポピュラスII	イマジニア	'92/8/28

(無作為抽出した1000通のハガキを集計)

ウワサのソフトウェア（海外編）

FLASHBACK

もう1年近く前に紹介した「OUT OF THIS WORLD」。完全ポリゴンにより，リアルな動きをする人体と背景，およびビジュアルシーンを質感レベルで完璧に統一し，あたかも映画の登場人物を操作しているかのように錯覚させた画期的な作品である。実際，僕にとっては衝撃的なゲームで，アクションアドベンチャー史におけるブレイクスルーとまで思ったものだ。

で，その第2弾「FLASHBACK」が登場した。ただし続編ではなく，別ストーリーである。舞台は未来。主人公は正体不明の敵基地からエアバイクで脱出したが撃ち落とされる。やがて主人公はジャングルの中で目を覚まし，唯一の武器である銃1丁を頼りに脱出行を繰り広げる。

人物の動きはさらに磨きがかかり，多彩かつ滑らかでシャープな動きをする。歩く，走る，跳ぶ，転がる，よじ登る。これらの動きが実に見事。コンビネーションも可能で，そのつながりも滑らかそのもの。助走をつけて谷間を跳び越え，間髪を入れずに前転，銃を構え，撃つという動作が，インターバルなしにできてしまう。操作を極めれば華麗な「見せる」プレイも可能。このへんは前作から正常進化したといえるだろう。とにかく操作そのものが楽しい。

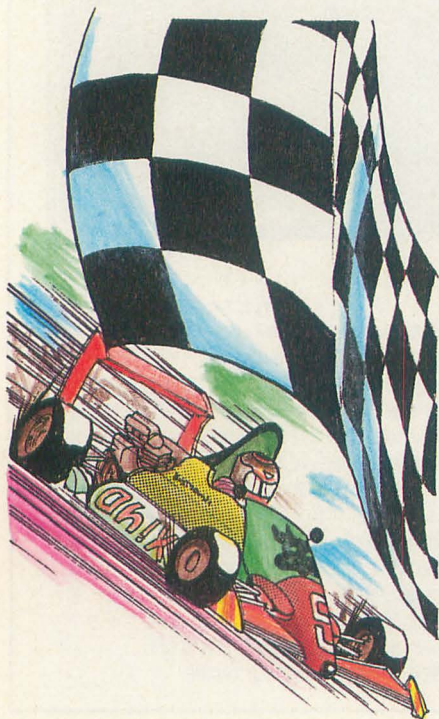
敵キャラクタも盛り沢山。ショットガンを撃ちまくる原始人風の敵，バックパックを背負って空中から主人公を狙い撃つ敵，加速装置を装備したサイボーグ，極めつきは攻撃時に人の形をとるアメーバ状の敵。それぞれ異なる攻撃方法でプレイヤーを苦しめる。動きもリアルだ。

半面，背景がペイント画になりビジュアルシーンが浮いてしまったとか，アイテム探しと敵の殺戮に腐心するゲームという印象が強くなったとか，説明を文章に頼るようになったとか，堕落ではと思う部分も多い。日本製ゲームの特徴を取り込み，いわゆるゲームっぽくはなったものの，前作で気に入っていたところの多くが失われてしまったのはちょっと残念だ。

ともあれ，機会があればぜひ見てほしいゲームである。アニメーションはもちろん，複雑な動作を簡単な操作で実現する手法についても勉強になる。このシステムなら「ダイ・ハード」も真にゲーム化できると確信する。また，国内機向けに移植されることがあったら，グラフィックの描き換えをしないことを望む。　(A.T.)

開発元　DELPHINE SOFTWARE

AFTER REVIEW

1992年Oh!Xゲーム大賞に輝いた「オーバーテイク」。期待が大きかっただけに辛口の批評もありますが，それもみな大きな満足あってのこと。寒い冬を吹き飛ばした熱いゲームに，たくさんの声が寄せられました。

オーバーテイク

▶僕はオリジナルチーム「SAZAESAN-RACING」のKATUO-ISONOでがんばってます。第2ドライバーはMASUO-FUGUTA，メカニックはタラちゃんとイクラちゃん。絶妙のコンビネーション。オーナー波平。メインスポンサーは海山商事。

大又　義嗣(18)大阪府

▶10MHzだと遅いけど，15MHz改造のPROなら遊べる。でも，NORMALモードで車をぶつけてしまう私にとっては，サイバーシステムを載せてほしい。

則包　晃弘(23)広島県

▶セナに当てるのが快感(おい)。

金子　孝康(19)茨城県

▶一般道を走っているときも同じようにブロックしたくなる。畑中　英喜(19)大阪府

▶実際の運転でやけにサイドミラーを気にするようになってしまった。でも私のは悲しいディーゼル。　安藤　道子(20)宮崎県

▶すごい。だけど頭上からのリプレイ機能が欲しかった。自分がどういう走りをしているかわからない。吉田　正行(18)愛知県

▶ベストラップをセーブできたらなおよい。

小山　優一(19)東京都

▶周回数100%でやったときに真の姿が見えてくる。大変だけど。

山下　智也(22)大阪府

▶まず，音がいい。セッティングが難しいが，それをテストランで最高のセッティングにしてレースするのがすっごく楽しい。

鈴木　幸太(19)神奈川県

▶ラスタースクロール方式としてはほぼ完璧な出来あがりをしている。が，ついでにX68000の限界も感じさせてくれた。

佐藤　靖(19)神奈川県

▶簡単すぎる！　処理が粗い！　ピットがヘン！　などいろいろ不満はありますが，まあ面白かった。次はがんばれ！

金渕　満(16)青森県

▶DAMAGEがついてしまったぁ～！　くそ～っ，よし最終コーナー。おっと減速だ。ガン。誰だ，後ろからぶつかったのは！　そのとき赤色のRETIREの文字が……。

松永　好司(17)富山県

▶ここまでやってくれたら何もいえません。X68000ユーザーはこのソフトハウスを大事にしないと罰が当たります。3Dスコープに対応していれば，完璧だった……。

西本　英樹(20)北海道

▶とにかく現実をリアルに再現しようとするこだわりを感じた。

辰己　真章(19)兵庫県

▶マシンのサウンドがたまらない。レインコンディションの雷鳴にエクスタシーを感じる。　田村　明広(27)神奈川県

▶なんてったって自分の好きなマシンに乗れるところがサイコー。

木村　匡志(21)千葉県

▶ドライブゲームとしては一級，いや超一

級だ。しかし，F1ゲームとしては……。あのピットインはいけない。なぜ突然スピードを変えるのだろうか。それからオーバーテイクの仕方である。後ろから追突すると抜けてしまう。普通その場でリタイアかかなりのダメージのはずだ。だが走れる。カーブの再現性にも不満はある。しかしハード的に無理があるのならせめてはじめの2つぐらい改良してほしい。ズームさんのアフターケアに期待したい。P.S.うまいスタートの仕方を誰か教えて！

山口　健児(17)鹿児島県

▶ZOOMレーシングチームで名前をNECO(36)，MODOKI(37)にすると顔が変わります(もどきは気持ちわるい)。

新井　信也(16)群馬県

▶ひさしぶりに手にまめを作ってしまいました。　増崎　達夫(28)神奈川県

▶デモのネコがラブリィ。

仲井　隆良(15)愛知県

▶F1マシンのセッティングが意外に面白かった。　伊藤　雅信(25)東京都

▶路上のオブジェがもっと表示できればスピード感がリアルになったのでは……。

橋本　和典(26)東京都

▶オープニングだけでひとつのソフトになりそう。　菰田　英和(22)神奈川県

▶面白い！　下手な私にもテストランという強い味方がいる(さすがに99周もやらないけど)。　来島　克樹(20)広島県

▶MIDIにたよらなくとも，すばらしい音づかい。　松坂　貴之(20)宮城県

▶プレイしてて車酔いした。

高田　真郷(23)北海道

▶データ量が豊富で，データを見るだけでも楽しめる。　新田　和夫(22)石川県

▶SC-55，MT-32，CM-64に対応しているし，曲もGood。オープニングが最高。

寺元　正(19)奈良県

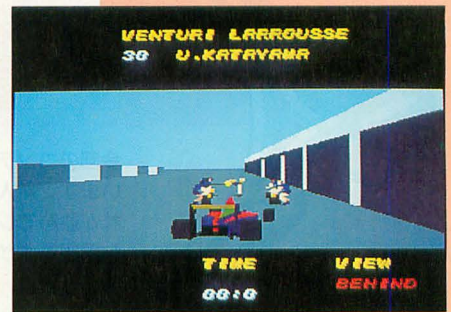
▶BGM，グラフィック，どれをとってもズーム味がした。CM-64がうなるぜ！ネコもうなるぜ！　X68000もうなるぜ！ごおー！　柳田　和宏（18）東京都

▶F-ZEROほどタイムアタックには燃えられなく，F1サーカス・ファミリーサーキットほどのスピード感もなく，ピットで半泣きになったが，リアルさは確かである。実名なので感情移入できてよかった。ヘナやナンセルとかだったら川にでも浮いていただろう。音楽はやっぱりズーム。オープニングは最高。　中島　貴史(16)滋賀県

▶左右ドライビングは少々軽くても，迫力があまり出てなくても，やっぱりあのデモとデータと多人数での遊びにピッタリだからよい。　武田　顕和(25)京都府

▶Oh!Xゲーム大賞受賞おめでとうございます。なんだかんだいっても，ユーザーに熱狂的な支持を受けるゲームを作ってしまうズームはタダモノじゃないのだ。

F1ゲームとして成立しているし，熱心に作ってもある。そりゃ期待していたものとは違っていたけれど，文句をいう筋合いはない。現在X68000で遊べる唯一のフルスペックF1ゲームという事実は大きい。写真うつり抜群，音楽もイイ。過剰なまでの演出。何よりオープニングが素晴らしい。

そこのけそこのけ××様のお通りだ。どかぬとOVERRIDEするぞ，とばかりに最下位からでも1～2周のうちに1位になれるゴボウ抜きの快感。そのあとも2位以下が食らいついて独走させない緊張感。順位を落としても絶望せずにすむし，トップでも楽勝にならない。エキサイティングなゲームを演出するための味付けである。

難度が少し低いかな。2シーズンやって2シーズンとも楽に圧勝できた。（A.T.）

発売中のソフト

★蒼き狼と白き牝鹿・元朝秘史　光栄　2/26
X68000用　5″2HD版　9,800円(税別)

★シムアント　イマジニア　2/26
X68000用　5″2HD版　12,800円(税別)

★幻影都市　ブラザー工業(TAKERU)　2/28
X68000用　3.5/5″2HD版　6,800円(税込)

新作情報

★メガロマニア　イマジニア　3/下
X68000用　5″2HD版　12,800円(税別)

★エトワールプリンセス　エグザクト　3/26
X68000用　3.5/5″2HD版　9,800円(税別)

★極　ログ　(「定石集」は発売中)　3/下
X68000用　5″2HD版　12,800円(税別)

★ヴェルスナーグ戦乱　ファミリーソフト　3/未
X68000用　3.5/5″2HD版　9,800円(税別)

★信長の野望・覇王伝　光栄　4/2
X68000用　5″2HD版　12,800円(税別)

★KU²　パンサーソフトウェア　4/9
X68000用　5″2HD版　7,800円(税別)

★餓狼伝説　ホームデータ　4/23
X68000用　5″2HD版　8,500円(税別)

★沈黙の艦隊　ジー・エー・エム
X68000用　3.5/5″2HD版　12,800円(税別)

★ロボスポーツ　イマジニア
X68000用　5″2HD版　価格未定

★Traum　M.N.M Software
X68000用　5″2HD版　価格未定

★鮫！ 鮫！ 鮫！　KANEKO
X68000用　5″2HD版　価格未定

★達人　KANEKO
X68000用　5″2HD版　価格未定

★エアバスター　KANEKO
X68000用　5″2HD版　価格未定

★サバッシュII　ポプコムソフト/グローディア
X68000用　5″2HD版　価格未定

★倉庫番リベンジ/SX-68K　シャープ
X68000用　5″2HD版　6,800円(税別)

★マージャンクエスト(仮題)　SPS
X68000用　5″2HD版　価格未定

★麻雀悟空・天竺への道　シャノアール
X68000用　5″2HD版　9,800円(税別)

★スターフォース　電波新聞社
X68000用　5″2HD版　4,800円(税別)

第5回アマチュアCGAコンテスト
入賞作品発表！

今回のコンテストは，応募総数が倍増して激戦となり，レベルもさらに上がっています。新人が多くなりましたが，なんだかんだいっても上位には，常連の顔ぶれが並んでしまいました。また，初心者向けに新しく設けられた1カット/4カット部門ですが，“どこが初心者やねん”というような作品も多く，侮れません。例年どおり，作品集のビデオも出しますので，ぜひご覧ください。

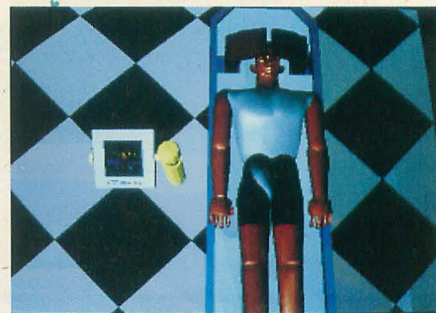
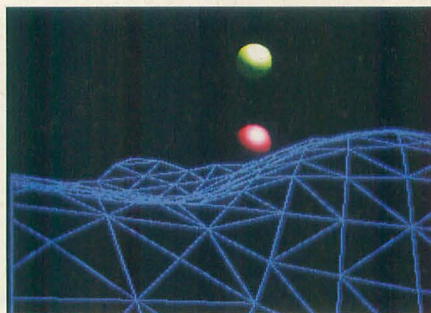
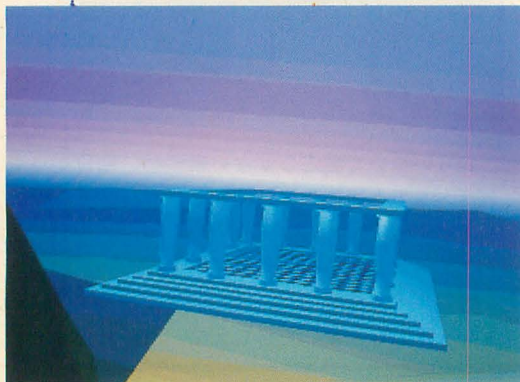
グランプリ　表彰状，賞金20万円

SWORD2

森山知己

2位以下を大きく引き離して，圧倒的なポイントでグランプリを受賞した，ものすごい作品。同時に，審査員誰ひとり絶賛しなかったという問題作でもある。アマチュアCGAの最高レベルとその問題点を同時に見せてくれる。内容は基本的に，一昨年のグランプリ受賞作「SWORD」のリメイク。

作者からのコメント

前回は画像の質まで手が回らなかったので，今回はCGAシステムのバージョンアップにともなって，少しグレードアップしました。

審査員の紹介

森啓次郎
「ASAHIパソコン」編集長。1971年，朝日新聞社に入社。「科学朝日」「週刊朝日」の編集を経て，1991年7月より現職を担当。

前田　徹
「Oh!X」編集長。ソフトバンクにて，シャープ系パソコン専門誌「Oh!MZ」編集長に引き続き，現職を担当。

太田　修
アニメ情報誌「NEW TYPE」編集長。週刊テレビ情報誌「ザ・テレビジョン」の編集にたずさわり，1990年ビデオ情報誌「ビデオでーた」編集長を経て，1991年4月より「NEW TYPE」の編集を担当。

寺島令子
ファミリーほのぼの4コマ漫画家。1979年「少年マガジンスペシャル増刊」にてデビュー。現在「ログイン」にて，「墜落日誌」を連載中。代表作は「くりこさんこんにちは」ほか。

塚田哲也
CGデザイナー，CGイラストレーター。1987年CGプロダクションJCGLに入社。現在はフリーで活躍中。CMの制作や個展を開くなど精力的な活動をしている。

古川タク
アニメーション作家，イラストレーター。1970年代後半より，コンピュータを使ったアニメーションの制作を始める。代表作は「驚盤」。

柴田忠男
「イラストレーション」編集部在籍中に，CGとスーパーリアルをえこひいきして，現在，コンピュータとデザイン，印刷を考える専門誌「スーパー・デザイニング」の編集長。「デジタル・イメージ」運営委員。

河野真太郎
1958年大阪に生まれ，アホとなり，東京に出て「ログイン」に入り，バカとなる。その結果，「ログイン」編集チョーとゆー，更生を受ける職に就く。

鎌田　優
プロジェクトチームDōGAスタッフ。1986年プロジェクトチームDōGAを設立。昨年，シャープ主催芸術祭グラフィック部門賞を受賞し，おおいに恥をかく。

作品賞　表彰状，賞金10万円

面会　客野　優

病院に２人の男(？)が見舞いにやってくる。彼の病を心配しながら……。客野氏はもう一作の「ハッピーバレンタイン」でも入選している。メッセージ性，テーマ性がはっきりとした作品だ。

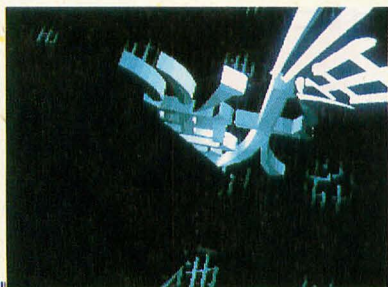
作者からのコメント

人に伝達することを仕事としている人間にとって忘れてはならないし，訴えつづけなくてはならないテーマだと思っています。

映像賞　表彰状，賞金10万円

日常的英雄綺譚

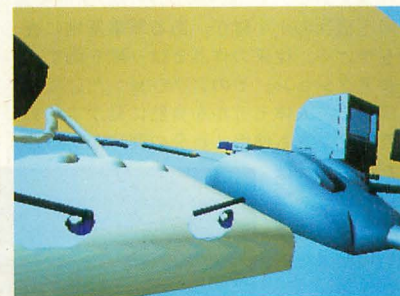
伊藤英基（HI-side）

勇者の父親の回想録。少し難解な作品ではあるが，元ネタを知っている人は何回か観れば理解できるだろう。画像はモノトーンに抑え，すべてのキャラクターを記号化して，その世界を表現している。このカッコよさは感動に値する。まさに映像詩と呼べる作品。３年連続しての映像賞の受賞。

作者からのコメント

見た目の派手さにとらわれず，過度な装飾を排することによって本質に迫ろうとした作品です。対象年齢 小学校３年以上。

努力賞　表彰状，賞金10万円

MOUSE

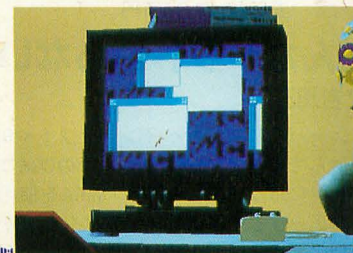
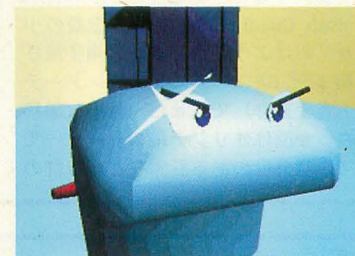
京大マイコンクラブ

Quick Timeで遊ぶMacintoshのマウスを，SPARCのマウスがあざ笑う。例年どおり，KMCが大量の人員，マシンを投入して制作している。物体変形の新技術も効果的に使われているし，ストーリーもわかりやすく，キャラクターの動きもかわいい。オチが決まっていれば，グランプリの可能性もあったかもしれない。

作者からのコメント

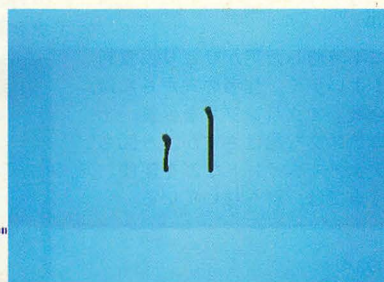
今年初めて物体変形に取り組みました。あたかもハンペンのごとく変形するマウス君たちをご覧ください。

アニメーション賞

表彰状，賞金10万円

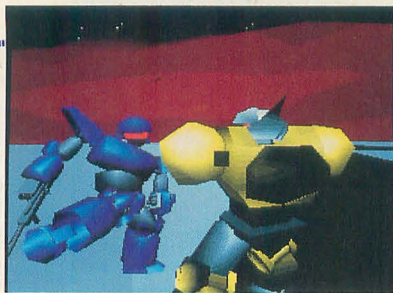
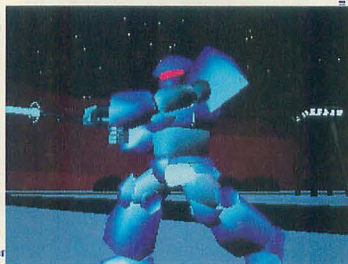
Complex　布山　毅

短い線と長い線。短い線はなんとかして長くなろうとするが……。Macintoshの「Macro Mind Director」を使って，即興的に制作されている。モノクロの２本の単純な線だけで，ちゃんとキャラクターを表現しているのがすごい。これこそアニメーションの原点といえる作品。

作者からのコメント

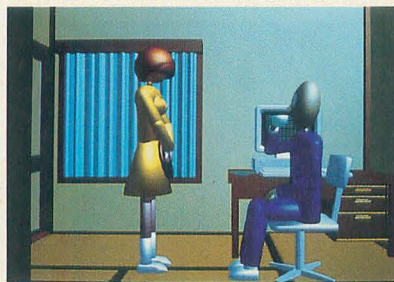
今回の制作で感じたのは，作ってすぐ観られる“即席”の面白さと，思いついたことをすぐアニメ化できる“即興”の楽しさ，つまり“創る喜び”です。

アクション賞
表彰状，賞金10万円

MISSION
浅野英史

降下機兵の１小隊が，ある軍事基地に夜襲をかける。従来の作品とは一線を画すバトルアクション。そのアクションだけで賞に値する。人体モデルを自然に動かすための自作ソフトも使用。自分の興味があることだけをすればよいという，アマチュアの特権がいい結果を生んだ作品。

作者からのコメント

モデリングが下手な分を，動きでカバーしようとがんばってみた作品です。ただひたすら“カッコいいロボットバトル”を目指しました。

佳作 表彰状，賞金5万円
ある夜の出来事
小島禎樹

今日も夜遅くまで仕事をする夫。それを心配する妻。そして，ある夜……。ハードはMacintosh Quadra700を使用。作者の小島さんが「おはようございますの帽子屋さん」（第３回CGAコンテストのエンターテイメント賞受賞作）で見せた，見事な編集力は健在。今回はオリジナルストーリーでの応募で，意外性のある展開などが抜群の出来。

作者からのコメント

短編を作ってみました。

佳作 表彰状，賞金5万円
こんなソフトいらない
立岩潤三

スーパーリアルなペイントソフトやRPGで遊んでいると……。いままでのCGAコンテストにはない変な作品。AMIGAを使用し，取り込みなどを駆使した表現はなかなか面白い。笑えるかどうかは，趣味の問題だろう。将来性を評価した。

作者からのコメント

４コマ的な笑える作品を作ってみたかった。タイムボカンシリーズのようなワンパターンの美学も少し取り入れたつもり。

佳作 表彰状，賞金5万円
A PLANET
宍戸光太郎

ある惑星に不時着した男がひとり。食料を求め，さまよい歩く。作者の宍戸さんは，昨年「猿蟹合戦」でグランプリを受賞。その画力，グロテスクな絵はさすが。今回もAMIGAでデラックスペイントなどを駆使。見せるという点では，群を抜いている。

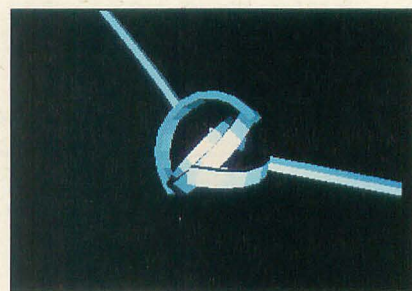
作者からのコメント

今年も手（マウス）描きのアニメーションです。今回は「気持ちいい動き」と「3D作品以上の奥行き」を表現することに重点を置きました。

佳作 表彰状，賞金5万円
MACHINE VISUALIZATION Vol.2
三ツ木淳

文章で書くとわかりにくい機械の構造を，CGAで解説していく。軸どうしでの動力のさまざまな伝え方などがひと目でわかる。今回の応募のなかで，最も楽しい作品のひとつともいえる。まさにCGAならではの作品であり，よい題材を選んでいる。

作者からのコメント

巧妙な機械を次々と創り出す人間の頭脳に乾杯！　単なる機械好きの趣味です。工夫がこらされた機械を考える楽しみが伝わればいいな。

佳作 表彰状，賞金5万円
COMPOSITION 001 “A”
鈴木陽二郎

ひと言でいうとわけがわからない作品。「ウゴウゴ・ルーガ」を見ているようで楽しい。MATIERなどを駆使して，１コマ１コマ加工することで，メタモルフォーゼなどを行っており，随所に面白い表現がなされている。CGAコンテストとしてはなかなか斬新な作品である。

作者からのコメント

さまざまな素材，イメージをコミカルにシュールに合成していく，それがこの作品の目的である。いちおう全体でひとつのテーマをもっているわけだが，あえて解説はしない。

入選 表彰状，賞金2万円

ハッピーバレンタイン

客野　優

たくさんのかわいいコックさんが集まってなにやら作りはじめた。さて，いったい何ができるやら……。作者の客野さんはまだ新婚で，夫にプレゼントする作品として制作したそう。色づかい，キャラクターデザインなどがしっかりできており，複数のコックさんが同時にワラワラと動くようすが楽しい。

作者からのコメント

CGA制作に取り組んでも，構想ばかりがふくらんで形にならない日々。とにかく，作品として仕上げることを第一の目標としました。

入選 表彰状，賞金2万円

Answer The Door

山畑裕嗣

何かが広場に落ちてくる。その広場にはたったひとつ，ドアがあった。最近ではめずらしい，PC-9801によるレイトレーシングの作品（「RAY-TREKII」使用）。単純なストーリーと画面構成が成功している。演出も面白い。

作者からのコメント

もともとは，「RAY-TREKII」を覚えるために作った作品ではありますが，努力はしたつもりです。まあ，楽しく作らせてもらいました。ちなみに，モーションはすべて打ち込み。疲れたー。

入賞 表彰状，賞金2万円

一件落着　川西翔也

腐敗した政治，駐車違反などの社会の問題点を，ドラえもんのようなアイテムで次々と解決していく。Macintoshを使用した２Ｄの作品で，手描きの水彩画のようなタッチが斬新だ。政府公報のような題材も変わっている。

作者からのコメント

自分がこの世の中で思うことを，素直に表現してみました（コミカルに）。このコンテストを知ったのが12月20日ごろで，あまり時間がかけられず，雑なところもありますが，よろしく。

入選 表彰状，賞金2万円

元気爆発 ジャンボルガー　西之園修

元気爆発ガンバルガーの発進＆合体シーンのパロディ。“アジオージャ”などで見せてくれたような，あいかわらず独自の世界をばく進する西之園氏のパワーには敬服する。技術的にもいろいろ凝った作品ではあるが，そういったことは気にせず，このバカバカしさに笑ってしまう。

作者からのコメント

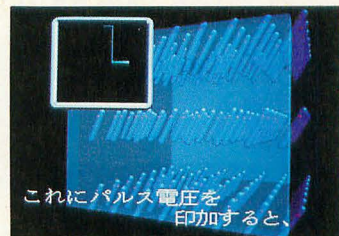
今年の作品は，ダークサイド部門に出品いたします（笑）。というのも，作品として成立していないと思いまして。まあ，内輪で見てください。

入選 表彰状，賞金2万円

FLC 強誘電性液晶　中尾健次

FLC液晶の原理を解説するビジネスプレゼンテーション。電流のON/OFFで，なぜ光を通したり通さなかったりするかがよくわかる。ビジネスプレゼンテーションも今後CGAが活躍しうるジャンルであり，興味深い作品だ。

作者からのコメント

初心者ですので練習作品として制作しました。モデリングが簡単で，動きも単純なので，すぐできるつもりだったんですが，結局１年がかりでした。いちおう，FLCの研究を仕事にしています。

第5回CGAコンテスト入選作品集の申し込み方

CGAマガジンのアニメーションを見ただけで，“スッゲー”などと驚いてもらっては困る。そんな方には，ぜひこのCGAコンテストのビデオを見てほしい。真のアマチュアCGAの最高峰だ。これを見ずして，アマチュアCGAは語れない。

そのパワー，楽しさは，ここに掲載されている写真（静止画像）からは絶対に伝わりきらない（百聞は一見にしかず？）。

ということで，申し込み方法は以下のとおりです。毎年トラブルが絶えず，スタッフ一同を悩ませていますので，くれぐれもご注意ください。今年はもうスタッフも宛名書きをしてくれないので，宛名シール（FDのラベルでよい）を同封して申し込んでね。

形態　VHSビデオテープ　１時間30分（βはありません）

配布価格　2,500円

（実費2,000円＋カンパ500円）

若干値上げしていますが，賞金総額，諸経費が大幅に増加し，コンテストの赤字がバカにならなくなってきましたので，ご了承ください。

期間　1993年３月15日～５月31日

発送　４月～７月

申し込み先

〒533　大阪市東淀川区淡路５-17-２　102号

DōGA内「５thコンテストビデオ係」

諸注意

・現金書留のみ

・自分の住所，氏名を書いた宛名シールを同封

・念のため電話番号明記

・“４thも送れ”“マニュアル送れ”など，ややこしい要求をせぬように

・CGAマガジンのユーザーは，以下のアンケートも同封してくれるとうれしい

1）　意見，感想

2）　アニメーションとデータベースのどちらを充実してほしいか？

3）　２枚組1,600円と１枚1,000円のどちらがよいか？

4）　希望するアニメーション

5）　希望するデータベース

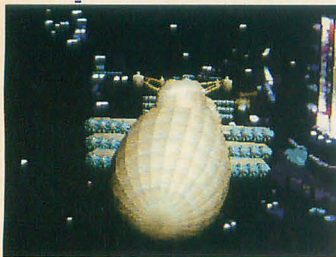
6）　システムとして使いにくかったところ

入選 表彰状，賞金2万円

X68000イメージデモ　北川博之

X68000のCM。昨年のX68000芸術祭用に制作したが，締め切りに間に合わなかったそうだ。作者の北川さんは，以前“手紙”という作品でも実写とCGの合成を行っていたが，今回も夜の街の実写を取り込んで合成している。CG部分もさることながら，この実写部分がよい雰囲気を出している。

作者からのコメント

本作品は，“X68000はこんなマシンなんだよ”というイメージから作成しました。芸術祭に間に合わず，そのままお蔵入りとなっていたもので，メモリ12Mバイトフル実装したX68000で再生できるようになっています。

入選 表彰状，賞金2万円

“D”　村松博之

宇宙から飛来した謎の球体“D”。地球防衛軍(?)の必死の抗戦もものともせず……。しっかりしたストーリーだが，展開や結末はなかなか奇妙である。モデリングやモーションデザインなどは，まだまだ初心者ながら，将来性を感じる作品だ。

作者からのコメント

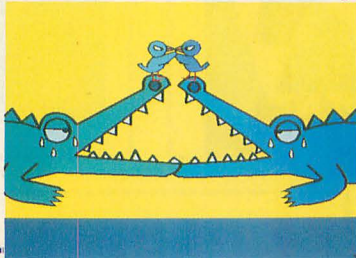
初めての作品制作でどれだけのものができるか挑戦してみました。とにかく，いままでにない“変”なものを作ってみたかった。

入選 表彰状，賞金2万円

CHUN-CHUN WORLD　佐野　元

かわいらしく，そしてときには意地悪な小鳥たちの絵日記。Macintoshによる2Dアニメーションで，作者の佐野さんはデザイナーを職業にしているだけあって，色づかい，画面構成などには安定した実力が感じられる。“オチ”もなかなかよい。

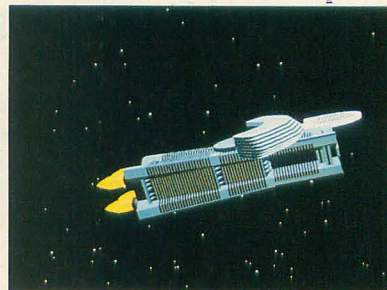
作者からのコメント

人間と同じように動物にも気持ちがあるに違いありません。鳥だって，ときに，意地悪だったり，照れ屋だったり。そんな世界を作ってみました。

参考入選 賞状，賞金2万円

SF大会

昨年行われた日本SF大会のエンディングアニメーション。SF大会の会場のビルが変形し，ロケットとなって宇宙に飛び立つ。演出など素晴らしい点も多いが，上映時間の大半がSF大会のスタッフのクレジットで占められており，一般作品としては認められなかった。

作者からのコメント

SF大会の会場そのものが，多くのSFファンと一緒に宇宙へ。臨場感，一体感を演出した映像作品です。

1カット部門賞 賞状，賞金1万円

OBJECT MECHANICAL HOUND
下岡正道

完璧な四足歩行モーションに驚嘆する。デザイン，モデリングも逸品。

メタリック人頭モデル
堀尾祐次

かつてないほどリアルな顔。作者自身の顔をサンプリングしている。

DRIVIN' WOMAN
由水　桂

映り込みの表現と自然な表情を持った人体モデルなど，高度な技術だらけ。

FREE WAY
土井輝久

モデリングもさることながら，光源と材質感の表現に徹底して凝っている。

OMDLデモ
砂岡克也

自作モーション記述言語OMDLによって，すべての物体を制御。

4カット部門賞 賞状，賞金1万円

ふしぎなえんとつ
藤村典由

モデリング，モーション，作品性どれをとっても，入選クラス以上。

Formula 1 Championship Machine
桑原正道

F1マシンの内部まで精密に作られた，ものすごいモデリング技術。

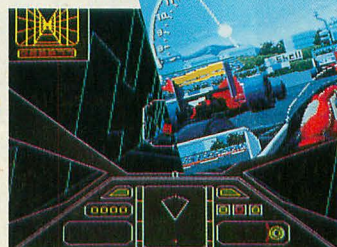
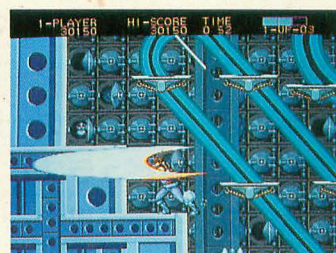
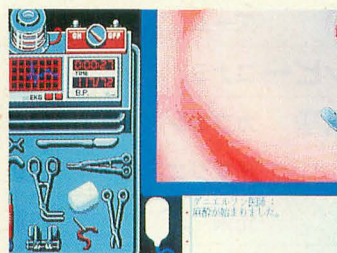
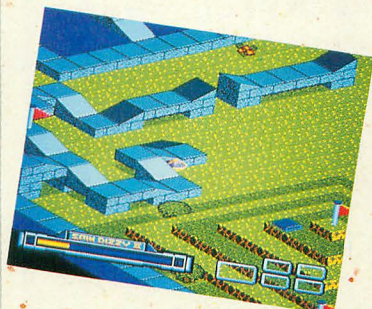
ZEPHYR
堀尾祐次

車のCM？　車のモデリングよりも，海を泳ぐシャチの雰囲気がよい。

［決定！1992年］

GAME OF THE YEAR

今年も数多くのおはがきどうもありがとうございました。いよいよ1992年GAME OF THE YEARの受賞作品を発表するときがやってきました。
1992年のX68000のゲーム界は，ビッグタイトルのニュースが絶えない，にぎやかなものでした。そのどれもが，プレイヤーの厳しい目に耐える一級のエンターテイメントばかりだったと思います。それだけに，そこからさらに厳しい目で審査され，晴れて受賞の栄誉を勝ち取った作品には大きな敬意が払われてしかるべきでしょう。しかもそれはプレイヤーである読者のみなさんの投票によって与えられるものであるため，それぞれの賞は一層の重みを増したものとなっています。
各賞の発表，Oh!Xスタッフがそれらの受賞作について語った座談会，スタッフそれぞれの個人的なベスト3などとあわせて，読者のみなさんももう一度，1992年のゲームを振り返ってみませんか。
そしてお待ちかね，ゲームへの言いたい放題，読者による「勝手にGAME OF THE YEAR」のコーナーには，買って遊んだユーザーだからこそいえる過激なコメントが満載。
あなたの投票した作品，お気に入りの作品の結果はどうだったのでしょうか。いよいよ1992年のGAME OF THE YEAR受賞作の発表です!!

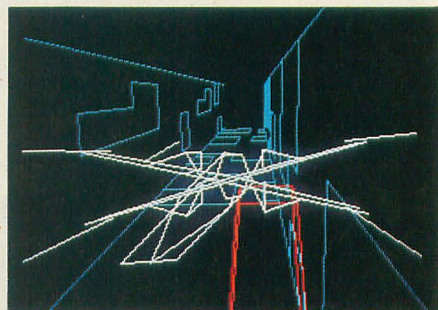
（編集協力：浦川　博之）

1992年 GAME OF THE

第1位 オーバーテイク　ズーム　222票

順位	タイトル	メーカー	票数
第2位	グラディウスII	コナミ	202票
第3位	ファイナルファイト	カプコン	133票
第4位	スターウォーズ	ビクター音楽産業	95票
第5位	ジェノサイド2	ズーム	71票
第6位	出たな!!ツインビー	コナミ	59票
第7位	ストライダー飛竜	カプコン	27票
第8位	ポピュラスII	イマジニア	26票
第9位	三國志III	光栄	23票
第10位	レミングス	イマジニア	18票

1992年のGAME OF THE YEAR，Oh!Xゲーム大賞はズームの「オーバーテイク」の手に！　ついにズームが初のOh!Xゲーム大賞受賞の栄誉に輝きました。ズームのスタッフのみなさん，本当におめでとうございます。
「ジェノサイド」の発売以来，常に注目を集めてきたズームが実は初受賞というのは，意外に思う人も多いことでしょう。卓越したグラフィックやサウンドの実力は高く評価されながらも，X68000のゲーム界では常にほかのソフトハウスと比較のうえで語られ，いわば準主役の座に甘んじてきました。しかし今年は，F1がテーマのレースゲーム「オーバーテイク」で一気にブレイクスルー，1992年のゲーム界の主役として実力と風格を強く印象づけました。

第2位　グラディウスII
昨年のGAME OF THE YEARで，このグラディウスIIでの受賞を予告(?)したコナミ。実力・評判ともに十分だったが惜しくも届かず2位。

第3位　ファイナルファイト
今年参入したカプコンの第1作。2人同時プレイも楽しいぞ。多彩な技を駆使して街のクズどもを蹴散らし，さらわれたジェシカを救い出せ！

第4位　スターウォーズ
昨年の2位に続き，今年も上位入賞を果たしたスターウォーズ。スピード感にあふれる画面と，多彩なトレースモードがセールスポイントだ。

この集計はOh!X1993年2月号のアンケートはがきより無作為抽出した1000件のデータによるものです。

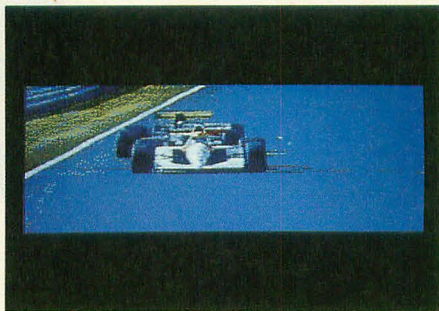
YEAR［受賞作発表］

この「オーバーテイク」が受賞作に輝いたのには，多くの理由が挙げられると思います。まず第一には「オーバーテイク」がX68000ユーザーに待ち望まれていたレースゲームであったこと。モータースポーツ人気の高まりとともに，レースゲームを求める声はずっとあったのですが，その期待に応えられるだけの作品はなかなか登場してきませんでした。そこに実力，ネームバリューともに誰もが認めるズームが，モータースポーツ人気をリードするＦ１を題材にゲームを作るというニュースが流れたのですから，発売前から過熱気味といえるほどの人気と注目が集まったのはしごく当然の成り行きでした。

ゲーム内容のほうも，ユーザーの期待にたがわぬものであったといっていいでしょう。「オーバーテイク」はＦ１のムード，レースの高揚感を画面上に十二分に再現しています。曲がりくねるコーナーを攻略し，タイヤの摩耗と戦っているうちに，あたかもＦ１のテレビ中継に入り込んだかのような錯覚を与えてくれます。画面に登場するドライバーやマシンは実在のものですし，データベースの装備やオープニングのこだわりぶりには「さすがズーム」とプレイヤーをうならせるものがあります。アーケードからの移植に頼らずオリジナルとしてゼロからゲームを作り始め，しっかりと「ズームらしい」ゲームを創りあげたその頑張りは，誰もが認めるところではないでしょうか。

集計当初から「オーバーテイク」と激烈な戦いを繰り広げ，惜しくも２位にフィニッシュしたのがコナミの「グラディウスII」

第5位　ジェノサイド2

多彩な動きと美しい画面がウリのジェノサイド2は，発売は比較的古くてもこの高人気。オーバーテイクよりズームらしくていいという声も……。

これだけでもソフトを買う価値がある，との声もあるデモ。演出力の高さをみせつけられます。

車のセッティングも，「Ｆ１に参加」という臨場感をかきたてるのに一役買っています。

です。１位と２位の票数がこんなに接近したのはいままでにないことですが，コナミのシューティングゲームの最高峰との呼び声が高い作品であり，投票ハガキにも気合の入ったものが多かったこの「グラディウスII」。２位の結果にはたくさんのため息が聞こえてきそうですね。原作の面白さは折り紙つきだし，移植の完成度，技術力も最高レベルです。こちらがOh!Xゲーム大賞だったとしても，首をひねるような人はいないでしょう。いずれにせよ，シューティングゲームが好きな人のおこづかいは，みんなコナミに行っちゃったんじゃないでしょうか。

３位にはカプコンの「ファイナルファイト」。カプコンの参入は1992年でいちばん大きなニュースに数えてもいいと思います。シューティングだけがアクションゲームじゃないとばかりに，格闘型アクション「ファイナルファイト」を発売，大反響を呼びました。ゲーム性や移植度はいうまでもなく完璧。X68000ユーザーのゲーム選びに，また魅力的な選択肢がひとつ加わることとなりました。

４位と５位は1991年の年末に発売されたソフトです。ここまで新鮮さを保ち，支持を集めたことはこれらの作品が話題性だけでなく高い完成度を備えていることを示しています。これらのソフトも含め，今回はどれもゲームの出来不出来に優劣のつけられないものばかり。昨年以上にレベルの高いところで戦いが繰り広げられた1992年のGAME OF THE YEARであったといえそうです。

Oh!Xゲーム大賞　受賞のことば
ズーム

このたびは，栄えある1992年Oh!Xゲーム大賞をいただき，スタッフ一同たいへん喜んでおります。

このオーバーテイクは，ズーム初のジャンルであったために，開発ツールやミュージックドライバなどすべてが一から作り直しで，試行錯誤の毎日。本当に完成するんだろうかと，ここ一年間，冷や汗ものでした。

いま振り返ってみると，信じられないことの連続でした。まず，マスターアップの半年くらい前に話していた「FOCA公認にしたいね」という冗談めいた話も実現し，さらにズームの５作品中いちばんの出荷数を記録。ゲーム大賞受賞にあたっては，もう夢のようです。

きっと最初で最後と思われるズームのラスターレースゲーム（いま現在，世界のレベルはたいへん高く，ポリゴン，ビットマップなどを駆使したレースゲームが大半を占めていますからね）ですが，必ずしもすべてにおいて満足のいく出来というわけではありませんでした。

しかし，スタッフ一同が全力でがんばった結果，こういった大きな賞をいただいたということは，ユーザーの方々に我々の熱意が届いたんだと自負しております。

応援してくださったユーザーの方々，ならびにシャープの方々，本当にありがとうございました。

グラフィック賞

1位 ジェノサイド2

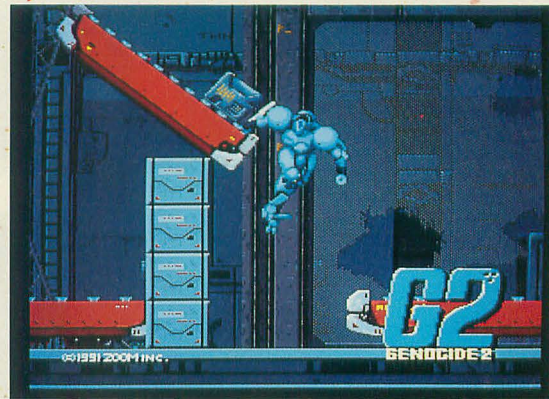
503票

順位	タイトル	票数
2位	ふしぎの海のナディア	170票
3位	オーバーテイク	46票
4位	シュートレンジ	34票
5位	バーンウェルト	32票

ノミネートの時点では「突出した存在はない」と申し上げたこのグラフィック賞ですが，見てください，この「ジェノサイド2」の圧倒的な得票数！　時期的な不利もなんのその，大勝利を収めました。

昨年も「ファランクス」で2位に入ったことからしても，ズームのグラフィックに関する実力はわかることと思います。特にこの「ジェノサイド2」は一連のズームの作品群のなかでも画力の集大成といった感があります。独特のくすんだ色調，統一されたメカデザインのセンス，それを迫力たっぷりに動かしてみせるアニメーション。すべての要素を使って，独特の「ジェノサイド2ワールド」を作り出すことに成功しています。

2位の「ふしぎの海のナディア」は，テレビアニメーションを制作していたガイナックスが作っただけあり，テレビ作品に非常に忠実なグラフィックを再現しています。原作のあるゲームの場合，自分の知っている世界やキャラクターに少しでも違和感があると，とたんにやる気をそがれてしまうもの。同じグラフィック，世界観で遊べる「ふしぎの海のナディア」はファンにとってはたまらなく嬉しい作品でしょう。

今回のこのグラフィック賞では，ノミネート作品以外のゲームがランクインするというハプニングが起きました。3位につけた「オーバーテイク」です。さすがズーム，1992年に占めるオーバーテイクの存在感の大きさを感じさせます。

受賞のことば　ズーム

ズームのグラフィックは，「こだわり」をもったデザイナーたちが連射マウス片手に描いております。妥協は許されません。特にジェノサイド2のときには，担当しているデザイナーの数が多かったので，濃厚なタッチと洗練された色遣いのグラフィックになっていると思います。

常に新しい表現方法や色の使い方などを日々研究しておりますので，これからも，ズームのグラフィックを堪能していただければ幸いです。

音楽賞

1位 出たな!!ツインビー

278票

順位	タイトル	票数
2位	オーバーテイク	245票
3位	グラディウスII	208票
4位	ジェノサイド2	111票
5位	スターウォーズ	16票
5位	スタートレーダー	16票

音楽賞でも一昨年末に発売されたゲームが頑張りました。コナミの「出たな!!ツインビー」です。前回は「パロディウスだ！」や「生中継68」の存在の大きさに，ノミネート作品からはもれていましたが，1992年になって，TOP10での息の長い人気をバックにつけてノミネートされ，見事に受賞を果たしました。

「覚えやすい」「クオリティが高い」「ゲームのムードに合っている」ということがゲームのBGMとしてプレイヤーに支持されるための条件だと思いますが，「出たな!!ツインビー」はそのどれもを非常に高い得点でクリアしています。加えてCM-64とSC-55に個別の対応をするなど，音楽にこだわるゲーマーの期待にいたれりつくせりの対応をしてくれました。

グラフィック賞ではノミネートされていないにもかかわらず3位に入った「オーバーテイク」ですが，ここでは逆にノミネートされていたにもかかわらず，惜しくも2位。エグゾーストノートの再生など，サウンドに対する取り組みは積極的でしたが，BGMの覚えやすさで「出たな!!ツインビー」に一歩譲ったようです。

しかしコナミとズームの一騎討ちとなったなかで，「グラディウスII」よりも前に発売された「出たな!!ツインビー」が「オーバーテイク」まで打ち負かして受賞するとは，「出たな!!ツインビー」がいかにプレイヤーの心をがっちりつかんだかということを雄弁に物語っていますね。

受賞のことば　コナミ

音楽賞に選んでくださった皆様，ありがとうございます。開発終了から1年半，これを担当したことが過去となりつつあるこの頃，嬉しい知らせに担当者一同狂気乱舞しています。開発当初は，3バージョンも入らないのでは，と悩んでいましたが，無事完成し，SC-55バージョンではアーケード版を超えた，と高く評価していただきました。そうやって生まれたX68000ツインビーの音たち。いつまでも可愛がってやってくださいね。

プログラミング技術賞

1位
オーバーテイク
369票

順位	タイトル	票数
2位	ストライダー飛竜	168票
3位	ファイナルファイト	158票
4位	バトルテック	58票
5位	グラディウスII	25票

ここでも「オーバーテイク」が圧倒的な支持を受け，プログラミング技術賞を受賞しました。いや，ホントに強い。

3Dタイプのカーレースゲームというのはそれだけでも技術力を必要とするものです。ズームはそこにさまざまな演出を盛り込み，あたかも車載カメラを見ているかのようなディテールの細かい画面を，しかも10MHzのX68000でもストレスなく遊べるように再現してみせました。また周辺の部分での技術にも目を見張るものがあります。通信対戦を可能にしたり，セッティングやピットインのシーンではポリゴンまで欲張ってしまうという技術の乱れ飛びが，ユーザーの心をとらえたのでしょう。

2位にはカプコンの「ストライダー飛竜」。いままで数多くのビデオゲームがX68000に移植されてきたために，つい完全移植が当たり前のように思えてしまいますが，よくよく見るとこの「ストライダー飛竜」，画面の半分もあるキャラクターがズンズン歩き回り，重力コントローラの周りをグルグルとスクロールしながら回ったりと，ハードな演出をこなしています。

3位の「ファイナルファイト」では，あえて再現しなかった部分もありましたが，投票結果はゲーム性を損なわない賢明な処置としてプレイヤー支持されていることを示しています。

おっと，5位には「グラディウスII」が乱入。みなさんコナミの技術力も忘れちゃいませんね。

受賞のことば　ズーム

プログラミング技術賞ですか！　すごいですね。

技術的にいちばん苦労したところは，やはりあのエンジン音でしょう。それと，敵車のパターン数は当初はもっと多かったのですが，2メガのメモリではちょっとつらかったので，泣く泣く削りました。

ラスター，PCM合成，MIDI，RS-232C通信，アナログジョイスティック……とにかく，処理が重いぃぃぃぃ！

ゲームデザイン賞

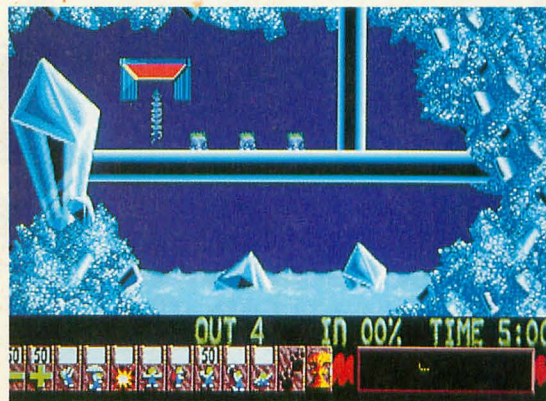
1位
レミングス
370票

順位	タイトル	票数
2位	ポピュラスII	190票
3位	ファーストクイーンII	66票
4位	キャッスルズ	55票
5位	スピンディジーII	49票

前回はシンプル・イズ・ベストの「ボンバーマン」が受賞したゲームデザイン賞ですが，今年は再び海外からの作品が脚光を浴びる年となりました。受賞に輝いたのはイマジニアのパズルゲーム「レミングス」です。

ちょこちょこと歩き回る可愛いレミングスを出口まで誘導してやる。ただそれだけのことなのですが，最初は簡単すぎると思っていたはずが気がつくと画面をにらんでうなっているという，バランスのとり方は魔術的。その辺のうまさは，遊び込んだ人ほど熱狂的に支持していることからもうかがえます。高度な技術がゲームの面白さを決定するのではないという，当たり前ながら非常に重要なことを教えてくれたあたりが厚い支持を得た理由ではないでしょうか。

2位の「ポピュラスII」も同じくイマジニアのゲームです。前作の「ポピュラス」も一昨年「シムシティー」に抑えられてプログラミング技術賞2位に甘んじています。残念ながら前作の仇をとることはできませんでした。もっとも，内容の濃さやプレイヤーを引き込む力に関しては「レミングス」と互角以上の実力を持っています。

海外からの移植ものが大勢を占めるなか，国産ゲームの「ファーストクイーンII」が健闘しています。自分のゲームスタイルを持っているソフトハウスに根強い支持が集まることは，ユーザーが話題作以外にもしっかりとした目を向けている証拠として歓迎したい傾向です。

受賞のことば　イマジニア

ゲームデザイン賞に選んでいただき，どうもありがとうございました。レミングスはいままでにないジャンルを狙って発売したため，この賞に選ばれたことは，我々にとってたいへん光栄であります。特にX68000版レミングスに関しては，他機種にない対戦モードを追加したこともあり，思い出深い商品です。これからも，この賞に満足することなく，大賞を受賞できるような作品を開発していきますので，応援よろしくお願いします。

座談会 1992

GAME OF THE YEARを語る

日頃，ゲームレビューなどでおなじみの，Oh!Xスタッフ5人による座談会です。1992年の受賞作を中心に，現在，そしてこれからのゲームの状況について各人，おおいに語ります。ゲームを愛するがゆえの辛口の分析，批評をまずはお読みください。

●

浦川博之(以下浦)　まず，大賞のオーバーテイクから話しましょうか。このゲームをどう感じるかはその人のゲーム体験次第って感じがしますけど。

西川善司(以下善)　一生懸命作ってるね。

丹　明彦(以下丹)　自分で作りたいものを作ったってのはいいですよね，なんか。

中野修一(以下中)　作りきったのかな？

浦　いや，今回は途中で路線変更があったみたいですけどね。

中　作った人は納得してるんでしょうか。

浦　ゲーム部分がすべてではない，という考え方だったら，すごく完成度が高いですよね。デモとかパッケージとか，お店で見て，買ってきて，立ち上げて，遊ぶまでのさまざまなステップを全体的にとらえてですね。

横内威至(以下横)　全体的にかっこいいF1ものを作ろうとしたってことで。

浦　フジテレビのF1中継から伝わってくるカッコよさをパソコン上で再現したかったというのが，デモにも出てますよね。だからおおげさな話，乗ってるドライバーの感覚がどうかってことは，きっとあまり関係ないんですよ。

丹　じゃ，やっぱり自動操縦モードが欲しいなあ。

浦　リプレイモードは……ポリゴンでないとできないか，それは。

横　いや，そんなことはないけど。

浦　違う視点からは見れないよね。

中　でもポリゴンでやったとして，ゲームとして面白くなるかというと疑問に思える。

浦　いや僕はこれはこれでいいと。ポリゴンでできなかったからこうなったと。

善　やりたいことにマシンが追いつかなかったってこともあるかもね。

浦　プログラミング技術のせいなのか，ハードの制約によるものなのか。

中　それができるハードってあるのかな。

浦　ポリゴナイザ搭載で68030を2基とか。

中　いや，何がやりたかったのかわからないので……。あの車の描き込み見たら，そんな甘いものじゃないでしょ。ポリゴンのステッカーもついてないマシンで満足するとは思えない。F1中継前のほとんどリアルタイムのレイトレーシングが必要ですよ。

丹　データベースモードに出てくる車くらいには妥協するんじゃない？

浦　あそこに出てくる車を走らせたかったんじゃないかなあ。

丹　あれはリアルタイムで動かすにはちょっと作りがていねいすぎるかもね。

横　ポリゴンでやってたら，完全にシミュレートしてガチガチのものをやってた。

中　それにしては，捨ててるものが多すぎるよ。

丹　なんで車26台を満足に管理できないかってのが……。なんでこんなに遅くなるのか。もっと遅い7MHzのAMIGA用ソフト

スタッフによる「勝手に」BEST 3

1位　レミングス
2位　ストライダー飛竜
3位　グラディウスII

「レミングス」はゲームデザインにまだ新機軸が出る可能性の残っていることを実感させる。アバウトで壮快な「ストライダー飛竜」の「玩具」発言は傑作だ。MIDI導入を決意させた「グラディウスII」の音楽。

海外作品では「WORLD CIRCUIT」(F1)と「PINBALL FANTASIES」(ピンボール)。ともにスタンダードとなるべき傑作。

業務用では「ヴァーチャ・レーシング」。シミュレーション性はいまいちだが，爽快感と演出は見習いたい。　(A.T.)

では現実にやってるわけだし。
中　計算してる量が桁違いなんですよ。オーバーテイクの場合少ないから。
善　PCMの割り込みで食われてたりして。
中　それはモードで変えられるでしょ。
横　グラフィックってそんなに速くないですからねぇ，X68000は。はっきりいっちゃえば非常にやりにくい。
浦　AMIGAはAV処理が魔術的に……。
中　速いのは確かですが，ものを動かすということに関していえば決してスプライトより速くはない。
円　スプライトの内部処理は単にレジスタのやりとりだけだから。グラフィックを使ったんじゃ10MHzのマシンで遅いのはしかたがない。
横　それでも，十分技術を使ってるように見えますけどねぇ。
中　それもあるけど，技術うんぬんの問題ではなくて……。まず，グラフィックでやったこと自体で相当重くなってますよね。描画自体はかなりギリギリでしょう。そこまでやるのに，26台走らせるのを捨てる必要があったのかとかですねぇ。
善　表示はせいぜい5，6台？
円　それ以上は絶対出ません。
横　ラスターものにしたら，敵とかはグラフィックに描く以外ないんじゃないですかね。だからこれはこれでしかたないと思う。
浦　まあ，マシンパワーが高くなったらまた違う作り方になるよね。
円　この段階で速いマシンに頼ってもらったんじゃ困るんじゃないでしょうか。
浦　まだできる，と。
中　ジェノサイド2を見てもねぇ。
浦　あれぐらいはできるはず？
中　スプライトをもっと使おうとは思わなかったのかな。
善　使ってるのはハンドルぐらいでしょ。
横　それは，優先順位の差なんですよね。単純にどこにどれを描くっていっても優先順位があるから。それで問題が出てくるんですよ。これでみるとグラフィックを最優先させなきゃならないから。
円　そこの違いか……。毎回グラフィックを表示しなきゃならないんですよね，ラスタータイプのゲームってのは。ポリゴンは，追いつかなかったら何フレームかあとに描き換えてもいいんだけど，これはとりあえず毎回描き換えてないとつじつまが合わなくなる。そういう計算してるから，と思いますけど違うのかな。
善　でも，車はスプライトにしてもよかった気がするな。ハンドルとかを全部グラフィックにして。256色モードだったら2画面持てるじゃない。
横　それが，だめなんです。スプライトの優先順位がBGと一緒になっちゃうんですよ。だから自分の車が常にいちばん手前にないとだめなんですよ。あれを全部スプライトで描くとバカですからね。自分をグラフィックで描いたとしても，それを手前に持ってきて，その向こうにBGで道路を表示してるってことは，さらにグラフィックで車を描かないと，道の上に車が出てこないんですよ。テキストはもう，背景オンリーですね。
浦　意外に画面割り当てが厳しいハードだったわけ？　X68000って。
中　路面をテキストにしたら？
横　でも，ラスターでアップダウンが出せないんですよ。BGじゃないとラスターでアップダウンまで処理できないんです。
浦　じゃ，このスタイルのゲームとしてはよくやっていると評価できる？
横　そうです。もう限界だと思います。
中　まだいけそうな気がするけど……。
善　タイヤの部分とか部分的にグラフィックに持っててっちゃって……無理かな？
浦　では，プログラミング的なことはそれぐらいにして，トータルディレクションというか，方向性みたいなものに関して……。F1のムードを味わうゲームであるという

スタッフによる「勝手に」BEST 3

1位　マチエール
2位　ムーンクレスタ
3位　PITAPAT

正直なところ，近頃X68000でゲームをするということはあまりなかったので，いきなりマチエールが1位になってますが……。しかしマチエールには「おおぅ，こんなこともできるんですかぁ!?」という喜びをいっぱい体験させてもらったし，2位のムンクレにも合体時に思わず「ピッピッピッ…」と呟いていた駄菓子屋ゲーム小僧だった頃を鮮明に思い出させてもらいましたし。3位は……個人的な思い入れの問題ですな。

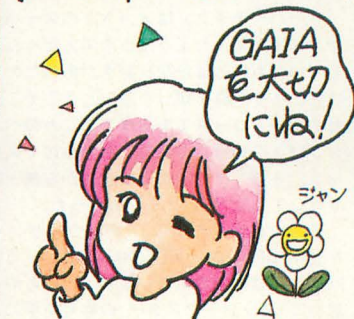
あと，去年よくやっていたゲームといえば「ワールドヒーローズ」に「龍虎の拳」，"ダッシュ"に「ぷよぷよ」といったところです。あとAMIGAのピンボール(タイトル失念)にもはまりました。こうしてみると我ながら対人間嗜好が高まっているようですね。やっぱり今年も対戦が熱い〜。
(哲)

スタッフによる「勝手に」BEST 3

1位　ストライダー飛竜
2位　ピットファイター(メガドライブ)
3位　超兄貴(PCエンジン)

やはりトップに挙げられるのは「ストライダー飛竜」である。移植であるとはいえ，ゲームとしての面白さはほかのオリジナルの追従を許さない。技術的にも，ROM吸い出しとしても確かなものがある。

次に挙げられるのはメガドライブの「ピットファイター」。やりこんだ奥の世界には，本来の遊び方ではないあまりに恥ずかしい世界が秘められている。敵同士の判定をフルに使い，敵同士を殴り合わせてやると，無能な敵は限りなく恥ずかしい勝負を見せてくれる。近くで動くものをなんでも追う，その知能レベルは，どんな生物にも劣る。取り込み画像でなければ味わえないエグい世界である。

最後はPCエンジンの「超兄貴」だ。あのクレイジーなセンスは，近代まれにみる恥ずかしさを表現しつくした。美醜の頂点を見せつけるこのセンスにはメロメロだ。　(横内威至)

ことについてはどうでしょう。
丹 X68000ユーザーでF1ファンってのがこういうものだという想定で作ってあるんだろうし，それは間違ってなかったんですよ。
浦 実際に，それについてはみんなすごく反応してますね。ただ，走って気持ちよければそれでいい，っていう人は満足させられなかった……？
横 そういう方向のゲームではないようですからね。
浦 F1のかおりをかぐソフトなんじゃない。
中 あれでは，デモの部分がどうしてもいちばん評価が高くなっちゃいますよね。
善 データモードとかね，ああいう電子辞書的なものプラスちょっとゲームで遊べるって感じかな。ゲームの比重って軽いように見えるんだけど。
中 周りの比重が大きいってのもあります。
丹 ゲームモードがついたマルチメディアソフトかな。

＊

浦 じゃあ，各部門賞に移りましょうか。
中 グラフィック賞のジェノサイド2は，やりたいことをちゃんとやってるじゃない。
善 自由奔放に作ってるよね。
浦 ジェノサイド2のほうが評価は高い？
善 ズームらしく，しかも100％何かやりとげたって感じがするじゃない。
浦 グラフィック賞とったのも，単に静止画がきれいってだけじゃなくてゲーム全体がきれいなグラフィック使ってるから。
中 もうアーケードレベルですからね。
浦 作風があるってのがえらいよね。個性がないメーカーも多いなかで。
ところで音楽賞は，オーバーテイクが出たな!! ツインビーに迫ってますね。出たツイのほうが評価はよかった。でも，同じはがきで投票してくるんだからこれはちょっと不思議。
中 だから「出たツイ」がとったってことはかなりすごいことですよ。実際のところオーバーテイクが音楽賞でも不思議はなかったんですが。内蔵音源に関する限りはオーバーテイクのほうがいいと思うし。
丹 オーバーテイクの音はゲーム中はあまり耳に入らない。
浦 ゲーム大賞の得票数を見て，みんなオーバーテイクで浮かれているのかなと思ったんだけど，音楽賞のほうを見たらそうでもないな，と。意外に冷静な読者。
善 オーバーテイクは音的には面白いけどね。あとで思い返して，あの曲がいいってのは出たツイのほうじゃないかな。1面と2面とエンディングがいいな，とかさ。
浦 レミングスは，「ゲームデザインといえばレミングス」っていうのは浸透したらしいけど，あまり売れなかった。
丹 人気が出なかったのがなんとも悲しいです，私は。
善 日本だけでしょ，売れなかったのは。
中 あっちではベストセラーですね。
善 国民性の違いか。
中 100パーセントおすすめできるかというと，ちょっと難しいかなというところはあったんですけどね。日本で受けるかということには疑問はあったんですよ。
善 日本って，こういう腰を落ち着けてマウスをカリカリやって楽しむパズルゲームってあんまり売れないじゃん。
横 そう，どっちかっていうとボタンバシバシのほうが。
丹 やたらうっとうしい面もあるし。
浦 でも，うっとうしいから嫌われたってわけじゃないでしょ？
善 地味だとか，つかみがなかったとか。
中 非常に美しい部分があるんですよ。どんどんのめり込む……最初のへんはね。
善 ネズミじゃなくてセーラー服の女の子だったら売れたかもね。1面クリアしたらビキニのネエちゃんが出てくるとか。でもさ，レミングスは丹さんが面白さを保証するって書いたよね。
丹 保証するよ。僕に説得力がないだけで。
浦 ポピュラスIIもがんばりましたね。
丹 本質的に新しいところがあったかといわれると……。結構，ファンが喜ぶような内容だとは思うけど。
善 宣伝や雑誌の取り上げ方もなんか地味

スタッフによる「勝手に」BEST 3

1位　三國志III
2位　卒業～Graduation～
3位　オーバーテイク

「三國志III」は遊べるゲームだった。MIDI対応など発展の余地は残るものの，ゲームシステムとしてはやりつくしたといえるだろう。ここいらで一発区切りをつけて，新たなるシステムを探求してもらいたくて1等賞にしておく。頼んだよ，コウ・シブサワさん。

「卒業」はX68000には発売されていない。私は研究室にあったハイパー国民機で遊んだのだけど，なかなか面白かった。右も左もわからずに育てていたら，一番期待の優等生がAV女優になっちゃうし。「プリンセスメーカー」でもいいけどね。X68000でもシムねーちゃんやりたいよ～。

「オーバーテイク」はメガドラのスーパー・モナコに毛が生えたような出来栄えだった。期待していたものとはかなり開きがあったので評価は分かれるが割り切って遊ぶとそこそこ遊べる。私のホームコースである南アフリカのベストラップは1分14秒11である。ちなみに1人用Qレース，5台，エフェクトは順位から無・無・有，サウンドはSC-55，タイプAだった。

最後に私事をすこし。「実はウィザードリィIV」(RETURN OF WERDNA)にはまっていた。X68000用は発売されていない。だけどやっとの思いで「グランドマスター」の称号を手に入れたので，嬉しくて書いてしまおう。でも，ウィザードリィシリーズをX68000に移植してほしいという期待も含めている。パスカルコンパイラがあるんだから，ちゃんと作ってよねSIR-TECHさん。　(S.K.)

スタッフによる「勝手に」BEST 3

1位　ファイナルファイト
2位　ポピュラスII
3位　グラディウスII

一応，順位は去年よく遊んだゲームをランクアップしてみた。「ファイナルファイト」は格闘ゲームが食わず嫌いだった私に新たな世界を垣間見せてくれた作品だ。「ポピュラスII」はとてもよくできたゲームだが，対戦モードが不完全でほとんどできず，前作のような対戦旋風は巻き起こらなかった。「グラディウスII」は私のなかで眠りかけていたシューティングスピリッツを呼び覚ましてくれた。

ここでX68000ゲーム界にとっての1992年を振り返ってみよう。去年発売されたゲームタイトルはほとんどがアクションゲーム系。アドベンチャー，RPGやシミュレーションの落ち着きのあるものが少なかった。

RPG，特にARPG系が少なかった。「サークII」ぐらいしか思い出せない。イースシリーズはI，IIIだけでIIがないという奇妙な状態。どうにかしてくれ。

シミュレーション関係は光栄，イマジニア，システムソフトとリリース元が確定している感があるが，それでも「バトルテック」や「ライフ＆デス」などの新規参入で新分野開拓の形跡もやや見られた。

アドベンチャーは壊滅状態。例年活躍していたリバーヒルが沈黙している。いったいどうしたというのだろう？ この分野はHソフトの天下となってしまったのか？ それならそれでもいいが。

さて，いわゆる大作といわれるアクションゲームもコナミ，カプコン，ズーム，電波新聞社とメーカーが偏っている。各ソフトハウスがリリースするゲームジャンルが確定しているのはユーザー側としては安心の反面，物足りない気もしてしまう。そういう点では，M.N.M.ソフトウェアが多種多様のジャンルで攻めてくるのでこれからも期待度満点といったところか。

超高速マシンX68030の出現で，今後X68000のゲームはどうなっていくのだろうか。030専用のゲームというものは発売されるのだろうか？ または，暗黙のうちにXVI以上の高速マシン専用のゲームなども出現してくるのだろうか？ 私はX68000ゲームソフト界にとって，1993年は「カオス」あるいは「ターニングポイント」の年になると予言する。　(善)

スタッフによる「勝手に」BEST 3

1位　グラディウスⅡ
2位　ファイナルファイト
3位　超人

別に無難な3本でまとめてもよかったのだが，カーレースのゲームはどうも嫌いなのであえて外させてもらった。グラディウスⅡやファイナルファイトは，そもそもがアーケードゲームの傑作ゲームである。それが，完全に近いレベルで移植されているのだから，これに関しては誰にも文句はいわせない。ジャンル的にも，シューテイングと格闘ゲームといったところで，両方買った読者の人も少なくないだろう。

そこで，あえて入れてある超人であるが，これはどちらかというと期待票の意味あいも多分に含んでいる。ゲームの規模がこぢんまりしているが，プレイしている瞬間の感覚を重視して作ってある点には，好感が持てるからである。若干ゲーム自体が古くさい感じもあるが，逆に重厚長大ゲームばかりがもてはやされるなかでは新鮮な印象を受けた。変に規模ばかり自慢するよりも，こういった小粒のオリジナルのよさも見直してもらいたいものである。　（八）

だったね。イマジニアも対戦大会やらなかったし。伸びなかったのは，対戦ができなかったから……？

丹　いったん対戦モードに入れれば，とりあえずゲームは進むんだけどね。

中　そのままやるぶんには問題ないんです。公平なマップを作ろうとして，それをセーブして一緒にロードして，さあ始めるとなると両方で別々にゲームが進行しちゃう。

丹　最初に対戦モードに入って，コンピュータが用意するマップだと問題ない。

＊

浦　では，1992年の傾向とかを総括してみましょうか。

善　会社一色，ジャンル一色。このあいだも誰かいってたけど，自然淘汰されて枝が全部切り落とされてこういう状態になったって感じですよね。

浦　昨年と比べると6位以下の顔ぶれが全然違う。いっそうアクション色が強くなりましたよね。X68000ユーザーは脳味噌が筋肉なんでしょうかね。

中　だって，出さないんだもの。

浦　出せば売れる。っていっておきましょうか。RPGとかアドベンチャーを出してくれるメーカーを期待したいですね。いまはコンシューマでも遊ぶものになってますけどねぇ。でも，レミングスはなぜか売れなかったし。

丹　なんとなく得体の知れないものには手を出さないんでしょうか。レミングスって，いままでの思考回路では……。とりあえずなんか心配じゃないですか。

浦　キャラちっちゃいし。

丹　慣れたゲームのほうが……というんではないでしょうか。

善　X68000版でも，探せば細かいのは結構いろいろあるんだけどね。

丹　でも衝動買いできない値段ですから。

善　そのへんがネックかな。

浦　衝動買いしても大丈夫な大作ソフトがそのときそのときのリーダーシップをしっかり握ってるから……。だから今年買ったソフトってどの人も似てたんじゃないかな。

善　コナミとカプコンがいろいろアーケードゲーム移植しちゃってるし。

丹　この知名度は大きすぎる。

浦　でもね，電波新聞社のがんばりは評価したいですよね。

善　これからどんどん出してほしいし。

浦　ターゲットを絞った企画って好感が持てるな。X68000ユーザーの全員には売ろうとしてないのがいいね。

＊

浦　では最後に，自分だったら，Oh!Xゲーム大賞に何に投票するかってのは？

善　僕は，やっぱりファイナルファイトかな。ああいうどんどん先に進んでいく格闘ゲームって面白いと思ってなかったんだけど，やってみたら結構よかった。効果音が面白いし。動きはストリートファイターIIなんかに比べるとちょっと粗いけど。いろんな技もあるし，いろんな敵も出てくるしね。絵もなかなかきれい。

横　ストライダー飛竜かな。結構適当にやってて遊べるから。自分で動いてて楽しいからね。

丹　僕は，大作の風格が足りないかなって思うけどレミングスを推します。

浦　そうか，風格が要求されるとこってありますね，やっぱり。オーバーテイクは風格があるからね。

丹　オーバーテイクをみんなが評価するのはわかる。今年のゲームっていえばやはりね。X68000らしいっていえば，そうなのかもしれないし。ユーザーも含めてね。

スタッフによる「勝手に」BEST 3

1位　グラディウスⅡ
2位　スターウォーズ
3位　ぷよぷよ（メガドライブ）

特に順位に根拠はないが，あえて順位をつければこうなるか。

今年度は自分で購入したゲームが2本あったが，1本は期待はずれで手放してしまった。残った1本が「グラディウスⅡ」。安全地帯まで再現してくれたコナミのプログラマ軍団の執念と，GS音源による広がりを持ったBGMには脱帽した。個人的にはOh!Xゲーム大賞は「グラディウスⅡ」に贈りたい。「スターウォーズ」は昨年のプログラミング技術賞受賞作だが，いまもってほかのゲームに見劣りしない素晴らしい動きをしている。徹底的に見せるプレイを楽しむことを教えてくれたソフトウェア作りに感動。美しいトレースプレイのために神経を減らし，効率的な敵の倒し方を研究する。最近はNIFTYなどの大手BBSに「スターウォーズ」をより楽しむためのパッチプログラムがアップロードされて，またまたM.N.M.スターウォーズの世界が広がった。

さて，「ぷよぷよ」は番外編となるが，いま最も熱くなっているゲームがこれだ。といってても対戦専用の接待ソフトとなっている。VSモードでは負けた人間が10秒以内にスタートボタンを押すと永遠にゲームが続くので「勝ち逃げ」はできない。この作り方がいやらしい。負けた瞬間にスタートボタンを押して，みんな「ぷよぷよ地獄」にはまっていくのである。　（影）

勝手に GAME OF THE YEAR

さあ，それでは1992年のゲームに対する読者のナマの声を見てみましょう。言いたい放題，好きなことを語る，ということで寄せられた声ですが，とにもかくにも読んでみて。異議のある方は……ゲームでもして，うっぷんを晴らしてくださいな。

親はなくとも子は育つで賞　シムアース

▶魚類の泳ぐ海を見ながら眠ってしまい，目を覚ますとそこには原子力文明の姿が。神(プレイヤー)の存在とはいったい……。

今野　清俊(20)山形県

ハードもないのに買っちゃったで賞　グラディウスII

▶1992年2月に購入。マニュアルだけを読むこと半年，9月にようやくXVIを買いました。

永尾　健次(24)大阪府

中年も楽しめたで賞　ふしぎの海のナディア

▶これまでゲームと名のついたものを最後までやったことのない私が，初めて最後まで楽しめた。おまけに夜の職員室でこっそりやっていたので，ついにおたくの仲間にされてしまった。

加藤　英輝(39)北海道

コンセプト賞　テラクレスタ/ムーンクレスタ

▶「移植」の意義をとらえた究極の企画だ。

小林　俊夫(20)埼玉県

目的は何だったで賞　バトルテック

▶みんなストーリーほったらかしで賞金稼ぎに没頭している。足をねらえ～。

古沢　達也(20)埼玉県

とにかくタイムアタックで賞　スターウォーズ

▶EASYモードで5分15秒92がベストタイム。1面はやろうと思えば45秒台は出せます。

田中　義章(18)北海道

買い損なったで賞　ファイナルファイト

▶評価が高そうなのでそろそろ購入しようと思ったら，どこにも売っていなかった(誰か譲ってくれ！)。

増田　秀樹(26)東京都

すごく値段が高いで賞　出たな!! ツインビー

▶CM-32Lの音で満足してたのに，GSの音を聴かされたら思わずCM-32Lを売っ払ってCM-500を買ってしまった。「出たツイ」9,800円＋「CM-500」115,000円＝124,800円って世界一高いゲームじゃないかしらん。

中田　健一(22)東京都

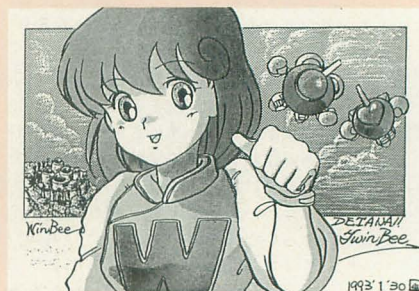
PCMに大爆賞　超人

▶アイテムを取ったときの叫び声に笑ってしまったのは私だけではないはず。新日本プロレスの橋本真也にジャンピングDDTをくらった感じである。

河本　直規(27)和歌山県

努力しているで賞　スピンディジーII

▶デモディスクが送られてきたのには驚きました。

内間　正晃(20)静岡県

オープニングがすごいで賞　オーバーテイク

▶あのオープニングをフロッピーディスクたった2枚だけでやっているなんて……。

小橋　玄央(19)茨城県

ハードディスクインストールしたいで賞　テラクレスタ/ムーンクレスタ

▶2本セットで4,900円のこのソフト，ノンプロテクトでもコ(ピー)せずにちゃんと買いますって。ねぇ，みなさん？

河野　太郎(19)東京都

PCエンジンへのアーケードゲーム移植作は高い確率で(当社調べ)X68000にも移植されるので次は……期待賞　コットン

▶理由は，読めばわかる。

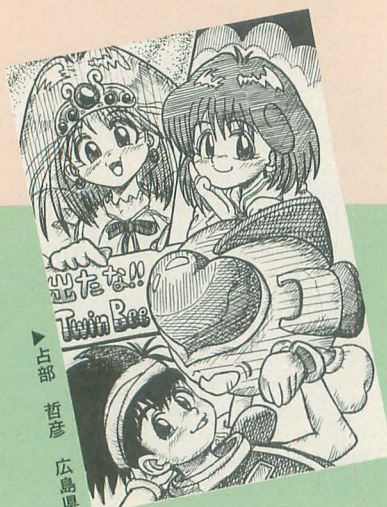
木村　行男(21)栃木県

どこに売っているので賞　ドラゴンストライク

▶とんでもない発想のゲームだと思うのだが，大阪・日本橋では見かけたことがない。ひょっとして関東地区限定発売とかだったりして……。

須賀院　隆志(21)和歌山県

▲山本　定男　東京都

▲枇杷木　亮　大阪府

▲占部　哲彦　広島県

▲横井　賢一　富山県

パソコンで出すんだから自主規制の葉っぱはなくてもいいで賞　ストライダー飛竜

▶わかるかな？　アマゾンの面だよ。

木島　智(22)宮城県

対戦が熱い！で賞
ファイナルファイトと出たな!! ツインビー

▶どちらも2人同時プレイのとき，画面上のすべてのキャラが敵となり，混沌としたプレイになるから。強敵(とも)を倒せ！

井上　敬介(21)神奈川県

前衛で賞　XENON2

▶あの面白さは，僕にしか理解できないだろう。　大島　大介(16)北海道

あたりまえで賞　ファイナルファイト

▶ガイが……いる。　佐藤　匠(21)埼玉県

我が前に敵はないで賞　チェルノブ

▶ナミダジョージョーのオープニング，エンディングがよい。やっぱし炭鉱夫。

藤本　格(22)神奈川県

また響子さんに会えるで賞　グラディウスII

▶去年の「パロディウスだ！」に続き，今年の「グラディウスII」。この人気は来年の「ゼクセクス」に持ち越しか!?　島本須美様カムバーック!!(笑)　小島　毅士(18)大阪府

驚異の40G賞　オーバーテイク

▶鈴鹿のシケインを280km/hで曲がると，これぐらい横Gがかかるはず……。

高井　智弘(19)大阪府

名前がまぎらわしいで賞　ライジングサン

▶ずっと「愚神礼賛」だと思い込んでいたから(笑)。　河野　裕文(17)静岡県

得したで賞&黙っていられないで賞　メタルオレンジEX

▶Aディスクのなかを見たら，Z-MUSICの最新版がマニュアル付きで入っていた。で，「得したで賞」。それから，18禁ソフトというハンデはあるもののゲーム(ブロックくずし)はとても面白く，本数の点で不利であるにもかかわらずX68000ユーザーのために発売を断行してくれたカスタムさんのために「黙っていられないで賞」。

桜井　直樹(29)神奈川県

▲小川　伸輔　宮城県

▲秋野　潤　静岡県

ムードを盛り上げるで賞　ムーンクレスタの音楽

▶ゲームが始まるときや，ドッキングのときの，あのやけにかん高い音楽(音？)を聴くと不思議とやる気が出てくる(ただうるさいだけだという話もあるが)。

黒武者　健一(23)神奈川県

1992年ベストソフトハウス賞　電波新聞社

▶ビデオゲームアンソロジーシリーズを始めた功績を讃えて。松本　拓司(18)埼玉県

ソフトハウスの鑑で賞　アルシスソフトウェア

▶いまさらながら「大戦略III'90」を買った。マニュアルがわかりにくかったのでアンケートハガキにその旨を書いて出したところ，それに対して謝罪の手紙と「スピンディジー」のオートデモが送られてきた。このときほど，マメにアンケートに答えてよかったと思ったことはない。最近X68000関係でぱっとした話を聞かないが，ぜひともまたユーザーの度肝を抜く派手な花火を打ち上げてほしいものだ。　鹿又　健　栃木県

とっても気持ちよかったで賞&ゲームのおまけ大賞　ふしぎの海のナディア

▶シューティングみたいに腕が痛くなるほどジョイスティックを動かす必要もなく，速度が遅くていらつくこともない。映画を観ているようで実に快適だった。また，結構量のあるマンガが付いているし，予約すると特製ディスクがもらえたりと，おまけが充実している。　青島　一高(24)静岡県

データベースはもはやホビーで賞　F-CARD GT

▶このソフトは私にデータを整理する楽しみを教えてくれた。青島　一高(24)静岡県

待ちくたびれたで賞　エトワールプリンセス

▶おお，いつのまにか「ストライダー飛竜」が私の手に！　片倉　純也(18)宮城県

▲姉帯　寛　茨城県

▲柿沼　光　埼玉県

▲岩瀬　貴代美　福岡県

1992年のゲームを盛り上げた
キャラクター

レミングス　レミングス

オープニングで可愛らしさをふりまいているレミングス。なのにいったんプレイヤーの手にかかると火に焼かれるわ，水に溺れるわ，ガケから落ちるわ，挙げ句の果てには爆破(！)されてしまうかわいそうな境遇のキャラクターです。しかし「爆発するレミングス」などと指定してくるところをみると，プレイヤーはあんまり「すまない」とは思ってないようですな。そのうち祟って出るかも。

ネコ　オーバーテイク

もはやX68000のゲーム界ではいちばんの有名キャラクターとなったズームのネコ。「オーバーテイク」では，実はV12エンジンを搭載していたという衝撃の事実(？)も明らかに。パフォーマンスもますます過激になる一方。コンフィグモードではあっちにこっちに大騒ぎの活躍をしてくれます。重量挙げにトライする姿がラブリーという声もちらほら。そういやネコモドキとの関係は依然として謎ですな。

Oh! My GODのオッサン
ファイナルファイト

Oh! My GODのオッサンとは，ボーナスステージでコーディたちに壊される車の持ち主らしき人物のこと。コーディたちがパーフェクトに車を壊したあとに現れ，ハラハラと大粒の涙を流します。このファイナルファイトのなかでいちばん損な役回り。彼を見るためにゲームをするという「熱狂的」ファンもいます。見たとこ子分のチンピラのようですが，そのわりにはホンダの旧型レジェンドなんかに乗っちゃって結構リッチな人かもしれません。まさか，誰かボスの運転手……？　だとしたら哀れさ倍増だぁ！　いまごろシティの海に浮かんでたりして……。

ええと，これまでは主演キャラクター賞と助演キャラクター賞を別々に紹介していたGAME OF THE YEARですが，読者のみなさんから主演だ助演だ主演助演だ，どれでもいいと多彩なおはがきをいただいたため，その声に応えるべく，企画を変更してお送りしてみました。

ということでして，以上，写真入りでご紹介いたしましたのが，まあ，お寄せいただいたなかでのダントツのベスト3であります。ちなみに，得票数は，

1位　レミングス　13票
2位　ネコ　6票
3位　Oh! My GODのオッサン　5票

でした。

では，これ以外の愛されたキャラクターたちをいくつか挙げてみましょう。

●ファイナルファイトの原始肉

このあたりは確かにありがたいですが。「画面内にあるから，と安心していてやられる自分」とは兵庫県の春名義行さん。

●ファイナルファイトの警官が投げ捨てたガム

なぜか体力回復になるそうです。細かいところをよく見てるね，どーも。エグさが人気のアイテムですが，「消化がよい」(！)という新説を唱えているのは北海道の太田哲雄さんです。

●ムーンクレスタの2号機

「理由は1号機が役に立たないから」(長崎県の関本正人さん)

●ムーンクレスタの3号機

「1号機と2号機のあとで役に立たないから」(栃木県の柿沼雄大さん)

う～む。2＞1＞3のヒエラルキーはムーンクレスタの基本ですな。

●ふしぎの海のナディアの生体兵器303号

「にせものだけど，にこにこ顔で料理するナディアが見られるのはコイツしかいない」(東京都の北浦暁光さん)。

なんだかナミダをそそるファン心理じゃあござんせんか，ねえ。

底抜け脱線？！
ゲーム体験談

ライフ&デス

石鹸で手を洗った。手袋もつけた。患部は消毒剤でまんべんなく拭いた。手術用の布を掛けた。抗生物質も投与してある。準備OKだうしししし。メスを持つ手が震えるぜ。うりゃ。

ギャ～～～～～ア～～～～～ア～～～～～!!!
なななななんだなんだなんだよぉ！
いきなりの悲鳴で，おもいきりびびる。
あ，麻酔を忘れてた。

このゲーム，難しい。今日も切るとこ間違えて怒られちゃった。それにしても，悲鳴が心地よい。わざと麻酔をかけずにメスを握るのが癖になりつつある……。

オレの使命はこの世から病人をなくすことだからなぁ。　山田　俊英(25)東京都

オーバーテイク～テストラン・モード～

●とにかくセッティングをいじくり，シケインを6速全開でクリアする。
●ウィングを0にして高速設定でコースをまわる。
●「妖怪通せんぼじじい」する。
●セナひとりをブロックし続けて，抜かれたらぶつける(ちなみに私はハッキネンを応援する)。　宮野　文武(20)神奈川県

オーバーテイク

トップをねらうことではなく，ブロックすることに命を燃やした！　なんとなく性格が知れそう。　永浜　貴光(21)千葉県

大戦略Ⅲ'90

添付マップをひととおり遊んだのでそろそろと思い，マップを作ってテストプレイ。初めは順調だったのですが，気がつくといつのまにか，首都に輸送艦が，海上に主力戦車が……。逆五芒星をモチーフにしたのがまずかったかな。

いろいろ調べた結果，4カ国でユニット数が1000を超えると発生することがわかった。　大塚　茂(22)神奈川県

1992年のゲームシーンを振り返る

「オーバーテイク」が2部門受賞，「ジェノサイド2」が500票獲得など，とにもかくにもズームの作品が強かったのが今回のGAME OF THE YEARの傾向です。これまでズームといえば，固定ファンの多さばかりがクローズアップされがちでしたが，今回の結果は人気と実力が広く通用する「本物」であることを実証しました。

「オーバーテイク」は待望のレースゲームということも人気の一因です。モータースポーツゲームの需要は大きいだけに，ほかのソフトハウスでも，努力を重ねて真のドライビングシミュレーションを作り上げてほしいものです。

そのズームを唯一打ち負かしたのが，意外にも「グラディウスII」ではなく「出たな!! ツインビー」。TOP10でも長く人気を維持した「出たツイ」の秘密は，どうやら2人同時プレイにありそうです。ただ，その人気のために，ゲーム大賞で「グラII」が票を集めきれずに惜しくも2位に終わったのは皮肉なことですが。

1992年のルーキー，カプコンは1年目ではやX68000用ソフトハウスの重鎮の座につきました。「ファイナルファイト」も「ストライダー飛竜」も特にプログラミング技術が高く評価されました。売り切れ続出でユーザーを広げきれず，ゲーム大賞の票が伸びなかったのが残念ですね。

1993年は，ズームの陰で涙をのんだこの2つのアーケードメーカーの巻き返しに特に期待がかかります。そのほかにもどんな作品が私たちの前に姿を現すのか，楽しみですね。

新学期 名作ゲーム再遊記

特別企画

時代が進むにつれて，新しいゲームが次々と生まれてくる。
そうして，ともすれば昔に買ったゲームは忘れられてしまう。
しかし，腑抜けな新入りを叩き落とし，自らを第一線から退かせないものたちもいる。
そんな激戦を生き抜いたものにだけ，名作という称号が与えられるのだ。
真に優れた老兵は死なず，そして，消え去りもしない。

スタークルーザー
アフターバーナー
パロディウスだ!
シムシティー
遥かなるオーガスタ
ねじ式
今夜も朝までPOWERFULまあじゃん2

スタークルーザー

Nishikawa Zenji 西川 善司

最近，アーケードやスーパーファミコンで，ポリゴン3Dゲームがもてはやされている。しかし，パソコンゲームにもポリゴンを使った名作ゲームはたくさんある。その代表格といえるのが，この「スタークルーザー」だ。

X68000用　3.5/5"2HD版2枚組　6,800円(税込)／ブラザー工業(TAKERU)　☎052(824)2493

夜，電気を消して，布団に潜り込む。目の前が真っ黒になって，なんだか宇宙にいるみたいな気分。そんな頭の中で思い描いた宇宙に，自分の空想の宇宙船を飛ばしたりすることってよくある。映画や小説，マンガ，いつの時代でも宇宙を題材にしたものは多いし人気もある。科学が進歩していろんなことがわかってきているけれど，宇宙に関してはまだまだ未知なる部分が多い。だからみんな宇宙に憧れるんだろう。

さて，そんなワクワクドキドキ(死語)な世界を冒険できるのが，アルシスソフトウェアの「スタークルーザー」だ。

不幸なスタークルーザーちゃん

この「スタークルーザー」は，1988年にアルシスソフトウェアから発売された3Dスペースアクションアドベンチャーゲームである。X1，PC-8801，PC-9801といくつかの機種で発売された。さらに，X68000，メガドライブなどにも移植され，同社で最も知名度の高いゲームとなった。

X68000版の発売は1989年。この年は「アフターバーナー」(電波新聞社)や「サンダーブレード」(シャープ)なども移植されて，X68000のゲームソフトに対して異常なまでの期待がかかった年であった。我々ユーザーがゲームソフトに対して，ハードウェア能力以上のことを望んでしまっていた年なのであった。不幸かな，そんなわけで「スタークルーザー」はナカナカの完成度を見せていたにもかかわらず，Oh!X誌上では結構キツく書かれてしまっていた。

ライバルであり，よき友でもあるギブソン

しかし，実際のところはかなりよくできた名作ゲームだった。たとえば，当時X68000では処理速度の点で難しいといわれたポリゴンで，複雑なモデルを実用的な処理速度で動かしていたし，発売されたばかりのアナログスティックにもちゃんと対応していた。ゲームを始めてみると最後までやりたくなるだけのストーリー展開の魅力は十分あるし，難易度バランスも悪くはない。宇宙モノでは不可欠ともいえる「広大さ」も見事に表現されていた(後述)。本当はもうちょっと高く評価されてもいい作品だったと私は思う。

クサくてシブイ

なんといってもアルシスのゲームのストーリーは「クサくてシブイ」のが魅力。同社の「リバイバー」(X1/PC-8801版)の頃からそうだった。なんかクスっと笑ってしまいそうなくらいクサいのだが，そこがなんともカッコイイのだ。

　　わざとらしい≦かっこいい
　　　　→クサくてシブイ

というような巧妙な不等式が，アルシスのゲームの魅力なのだ。

「スタークルーザー」について何も知らない人もいると思うので，ちょっとプロローグをお話ししよう。

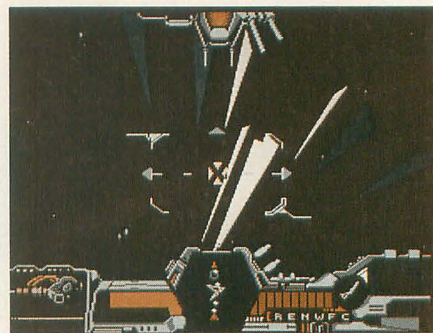
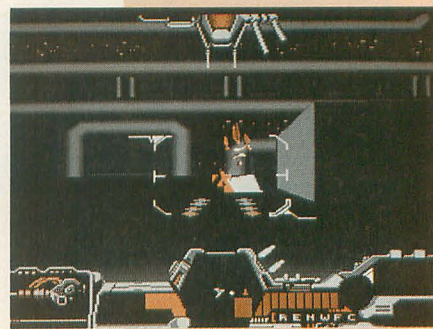
「スタークルーザー」の時代背景となるのは，人類が異星人文明との接触を果たし，宇宙に進出している遠い未来だ。

25世紀，さらなる人類文明の発展を願った人類は銀河連邦に加盟しようとする。このことをよく思わない排他的な思想をもったものが，連邦の勢力が及ばない辺境地を中心にテロを働くようになってしまった。テロリストたちには賞金がかけられ，そして，このテロリストたちを排除することを生業とする命知らずの男たちがいた。彼らは「ハンター」と呼ばれ，そのハンターのひとりをプレイヤーが演じることになるわけだ。ちなみにプレイヤーの名前はブライアンだ。群れをなさない1匹狼のハンターである。くー！　カッチョイイ。

ブライアンはあるテロリストを深追いしすぎ，巨大テロリスト組織VOIDの基地の勢力圏に迷い込んだ。それどころか，テロリストに自機を撃ち落とされてしまう。不時着したところがなんとVOIDの基地！　ゲームが始まるなりいきなりピンチだ。

どう？　なんかプレイしてみたくなってきたでしょ。

どんなゲーム？

ゲームの舞台は宇宙空間と惑星上のダンジョンの2つ。登場人物からミッションを頼まれたり，テロリストと戦って得た賞金

宇宙は広い

基地内での戦闘シーン

で自機を改造パワーアップ，新兵器を装備したりと，突き詰めて考えるとごく普通の3DダンジョンタイプのファンタジーRPGとそう大差はないといわれそう。ただ，全編ポリゴンで描き出される敵機や背景，3Dシューティングによる戦闘シーンはそういったことを気づかせないような魅力とオリジナリティがある。

ダンジョンの中の敵機は乱数で発生するようなものではなく，あらかじめ決められた出現位置にのみ出現し(レーダーで確認できる)，撃退すれば二度とその場所には出現しない。ゲームバランスを考えた設計だ。これによってダンジョン内で迷ったときにも理不尽に敵になぶり殺されることはないし，一度倒した敵はマップから消えるので，レーダー中の敵の座標から現在位置が未踏かどうかも判断がつく。このあたりは「ゲームの都合による，世界のデフォルメ」といった感じで賛否両論だろうが，結果として遊びやすくなっているので，これでよかったと私は思う。

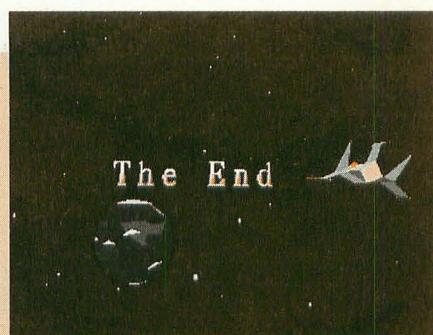
宇宙は広い

宇宙は誰がなんといおうと広い。だけれど，ゲームでそれを表現するのは非常に難しい。

もう一度いう，宇宙は広い

君はエンディングを見たか？

たとえば，シューティングゲームでこれを表現できたゲームがいくつあっただろうか。美しい宇宙が描かれた背景のゲームはいくつもあっただろうが，「広さ」を表現できていたゲームはそうはなかったと思う。

ゲーム開始当初は，プレイヤーはたいしたワープ装置も持っていないので，星間移動は通常空間を巡航しなくてはならない。この移動中に隕石地帯に突入したり，指名手配中のテロリストと遭遇することもあるわけだ。燃料が少ないときや，シールドパワーが低いときの敵機との宇宙戦は，かなりの緊張を強いられる。この緊張の連続の末，やっと目的の星に着くわけだが，この航行時間と敵の出現率がなかなかいいバランスを醸し出していて，プレイヤーは本当に「宇宙は危険でいっぱいだ」という気持ちに満たされる。

さらに，星系は太陽系だけではなくいくつかの星系があって，この星系間の移動には重力カタパルトという宇宙港を利用する。このあたりにも「スタークルーザー」の世界のスケールの広大さが表現されている。

また「スタークルーザー」では宇宙をX，Y，Zの3次元座標で区分しているのだが，この座標もひと役買っている。

たとえばゲーム中，「××星系の座標○△□から救難信号」といった緊急無線が入ったりして，その座標へ向けてマイシップを飛ばす気分は，もう主人公ブライアンそのものだ。

心に残ったゲームは?

私はアーケードゲーム基板を所有するほどのゲーム好きだが，ご家庭用ゲーム機は持っていない。というわけで，私が好きなゲームはみんなパソコン用ばかりだ。ちょっと挙げてみよう。

●MZ-700編

タイムシークレット(BOND SOFT)
タイムトンネル　(BOND SOFT)

古いパソコンゲーマーならば誰でも知っている，アドベンチャーゲームの名作だ。MZ-700のセミグラフィックを極限(?)まで駆使した美麗な画面，そして当時のゲームシナリオとしては先駆的なタイムパラドックスは絶賛された。続編の「タイムシークレット2」は「タイムトンネル」というタイトルで発売され，こちらもなかなか高度な時間トリックを扱った優秀な作品だった。そしてパート3は「近日発売予定」のまま消えてなくなってしまった。現在，BOND SOFTはどうなっているのか？　作者のネコジャラ氏の消息は？

●X1編

ユーフォリー(システムサコム)

ちょっとおふざけ気味のストーリーと登場キャラクターたちがかわいいアクションRPG。主人公の冒険の動機と目的が「母親の病気の特効薬探し」というのが目新しかった。故斎藤学氏の歌えるゲームミュージックに乗って，広大なマップを旅した日々は忘れられない。

ザナドゥ(日本ファルコム)

アクションRPGの古典的名作。トリッキーなマップを攻略していく快感は，いまでは遥かなる思い出。当時，私は若かった。カルマっていうパラメータが人泣かせだったなあ。

リバイバー(アルシスソフトウェア)

「おりゃー！」

わかる人だけ笑ってください。

●PC-8801編

ソーサリアン(日本ファルコム)

このゲームをしたいがために，私はPC-8801FHを買った。あとからX1turbo版が出た。悲しかった。

●PC-9801編

獣神ローガス(ランダムハウス)

長い間待たされたのに，発売されたのはPC-9801版のみ。でも，時間がかかっただけあって，ロボットもののアクションゲームとしては最高峰の出来に仕上がっていた。私はこのゲームのためにPC-9801互換機のPC-286VEを買ってしまった。自分以外にも僚機が戦ってくれるというアイデアに脱帽。アニメが好きな友人にいわせると，「このゲームはボトムズとレイズナーの世界を味わえるよね」だそうだ。

●X68000編

スタークルーザー(アルシスソフトウェア)

本文参照。

ポピュラス(イマジニア)

ポピュラスはコンピュータの中でしかできない，完璧なゲーム性をもった仮想現実ゲームだ。パソコンを使っての対戦ゲームの面白さを，世間に知らしめた作品だ。

スターウォーズ(M.N.M Software)

同名映画のクライマックスをワイヤーフレームで余すことなく再現。普通にゲームをしても楽しいが，ライバルがいればタイムアタックが熱い。ゲームをプレイしたあとにもお楽しみが待っている，そんなサービス精神にも歓喜。

ノスタルジア1907(シュールド・ウェーヴ)

練りに練ったストーリーと2重3重のドンデン返し，キャラクターの魅力，セピア調の美しいグラフィック。マンネリ化するこの手のジャンルのアドベンチャーゲームに，一石を投じた作品。

アフターバーナー

Izumi Daisuke 泉　大介

アーケードゲームにもいろいろあるもので，簡単にX68000へ移植できそうなのもあるし，絶対できそうにないのもある。しかし，そうした予測を覆し，生まれでた名作はたくさん存在する。このゲームもそのひとつだ。

X68000用　3.5/5"2HD版2枚組　9,200円(税別)／電波新聞社　☎03(3445)6111

1987年，パソコンユーザー皆がその目を疑うような事件が持ち上がった。それは1985年にアーケードゲームフリークを襲ったのと同じ，あるいはそれ以上の驚きだったに違いない。X68000にセガの珠玉の名作アーケードゲーム，「スペースハリアー」が移植されたのだ。

「スペースハリアー」は疑似3Dを使った前方向(?)スクロールシューティングゲームである。主人公ハリアーを駆って，画面の奥へ奥へと走り，飛翔していく。迫りくる敵の質感，躍動感溢れるBGM，スタート時の緊張をいや増す「Get Ready!」という掛け声，そして，なによりZ軸スクロールのスピード感。「スペースハリアー」はそれまでのゲームの常識が通用しない，時代を超えた逸品だったといえるだろう。

疑似3Dは，大小数パターンのグラフィックデータを順次画面に表示していくことで実現している。データ量や作成にかかる手間の制約があるため，全パターンをもつわけにはいかず，特徴的なパターン以外はその場で作成されることになる。つまりグラフィックパターンの拡大縮小が必要なのだ。これは専用のハードウェアを備えたアーケードゲーム機でなければ，とてもできるものではない。誰もがそう考えていたし，そう信じてもいた。

それがパソコンで動く！　それも，アーケードゲームの雰囲気をまったく損なうことなしに。驚愕のあまり腰を抜かしたゲームフリークがいたとしても，決して不思議ではない。

ロックオンした敵にミサイルを撃ち込む

ぐるりと1回転するのは実に気持ちいい

アフターバーナー登場

そしてここに紹介する「アフターバーナー」は，「スペースハリアー」の2年後にセガが発表した，3Dシューティングゲームの第2弾である。今回プレイヤーはF14トムキャットを駆って，大空を飛びながら空中戦を行う。

「スペースハリアー」では，主人公は画面を上下左右に動き回って敵を撃ち落とすのだが，地平線は固定されていた。今回は違う。F14トムキャットが右に左に旋回すると，風景がそれにあわせて右上がり，左上がりに傾くのだ。それもZ軸のスクロールを続けながら，である。

きわめつけは空中回転だ。素早く操縦桿を動かすことで，F14トムキャットをクルリと空中回転させることができる。ところが，このときF14トムキャットはある程度以上は傾かない。その代わりに，風景のほうがぐるりと1回転するのである。もちろん，Z軸スクロールを続けながら，である。

さすがアーケードゲーム。X68000版「スペースハリアー」の移植によって，3Dスクロールシューティングゲームに目覚めたパソコンユーザーも，これは無理だとあきらめていたのではないだろうか。

無理やり移植することはできても，きっとゲームはそれなりで，アーケードゲームのような爽快感は得られないに違いない，と思っていたことだろう。事実，先行したFM TOWNS用の「アフターバーナー」は，いかにも移植しましたというレベルにとどまっていた。

そんなユーザーの不安を呵々と笑い飛ばすかのように，質感，スピード感，躍動感，いずれをとってもアーケード版に遜色ないレベルで，X68000版「アフターバーナー」は登場した。1989年のことである。プレイ中の画面を見ても，誰もそれが10MHzの68000で動いているとは思わないことだろう。まさにソフトウェアの勝利である。

空中給油や着陸補給シーンも忠実に再現

アフターバーナーはこんなゲーム

F14トムキャットを操作して，迫りくる敵を殲滅(せんめつ)する。それが「アフターバーナー」の目的だ。迫りくる，というところがポイントで，基本的に敵の迎撃は前面から行う。もちろん，マッハ某で飛んでいる戦闘機どうしが正面から攻撃しあうなんてことは，実際の戦闘ではよほどの幸運がないかぎり無理な話だ。互いの操縦技術を駆使して後ろを取り合うというのが本来の空中戦の醍醐味であり，フライトシミュレーションベースのゲームなら間違いなくそうなる。

「アフターバーナー」はシューティングゲームなのだということを忘れてはいけない。そうでないと，どうしてとんぼ返りできないんだとか，俺は引き返したいんだとか，どうして前にしか飛べないんだとか，貴ノ花の婚約破棄劇はクサすぎるとか，やはり

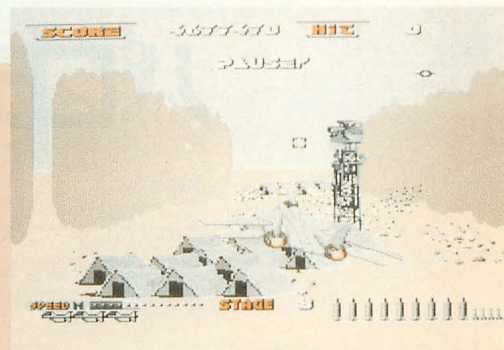
背後に敵が近づくと声で教えてくれる

ボーナスステージもちゃんとある

裏に何かあるんじゃないかとか，そもそも国会議員に自分で自分の首を絞める政治改革などできるわけがないじゃないかとかいった枝葉末節が目についてゲームどころではなくなる。したがってゲームの目的もいい直す必要があるだろう。敵を殲滅するのではなく，攻撃をかわして敵を撃墜しながら，どこまで生き残れるかを競うのだ。

迫りくる敵を撃墜するための武器は2つある。ひとつは空対空ミサイル。もうひとつはバルカン砲である。

ミサイルは自動追尾式になっており，画面に表示されている照準を敵機に重ねるとロックオンするようになっている。と同時に，「FIRE！」の掛け声。一定時間自動的にロックしつづけるので，照準で編隊をすっとなぞれば，4,5機まとめてロックオンすることも可能だ。ミサイルボタンを押したときの掛け声が，「ファファファ，ファイヤー！」とまた楽しい。

発射ボタンを連打すると，ミサイルが白い煙をたなびかせながら，それぞれの目標に突進し撃滅する。アーケード版ではこの煙がベタだったのが，X68000版ではメッシュパターンになっている。これはハードの制限からだが，向こうが見やすくなっているのでかえってうれしい。

ハードの進化を促した

「アフターバーナー」のユニークな点は，シューティングゲームの定番アイテムともいえるジョイスティックを拒否したことである。ご存じのように一般のジョイスティックやジョイパッドは，レバーを上に入れた/入れていないという2値でしか情報を伝えることができない。照準で敵機をなぞるには，これではいかにも役不足だ。

代わって「アフターバーナー」が採用したのは，なんとあろうことかマウスである。しかもこれが，実にしっくりとくる。マウスで戦闘機を自在に操ることができるなど，いったい誰が想像したろうか。アナログ(的な)データを得ることのできる，マウスの特徴がいかんなく発揮される。ちなみに空対空ミサイル発射は左ボタン，バルカン砲発射は右ボタンである。バルカン砲には連射機能がついているので，マウスのボタンを傷める心配はない。とはいえ，ゲームともなると結構乱暴にマウスを扱うので，かなりのユーザーがマウスを買い替える羽目になったということだが。

そして忘れてはならないのが，シャープからこの「アフターバーナー」のために発売された(?)アナログスティックである。戦闘機の操縦桿を模した作りの本格的なもので，普通のジョイスティックとしても使える(「オーバーテイク」用のハンドルアタッチメントはまだか)。エポックメイキングなソフトは数々あれど，ハードまで作らせてしまったソフトはこれくらいだろう。

ほしいゲームのその先に

ひと目見たその日から，ほしくてほしくてたまらなくなるゲームがある。そのゲームをプレイするためなら，マシンを購入してもいいとさえ思ってしまう。「そんな動機で散財するなんて。子供じゃあるまいし」と理性はブレーキをかけようとするが，どっこい，感情のほうもなかなかに頑強だ。「こんなに面白いゲームをあきらめるのか。いまお前が持っているマシンではプレイできないのだぞ」と揺さぶりをかけてくる。

人は常に，理性と感情の鬩(せめ)ぎあいの狭間に揺れている。たいていの場合はそれまで過ごしてきた人生の経験から反射的に取捨選択しているため，この葛藤が意識の縁に上ってこないだけである。美人だ，でも声をかけるのはよそう(これまでうまくいったためしがないから)。ステーキが食いたい，でもハンバーグにしよう(給料日前だから)。

人生は理性と感情の二極の間を行ったり来たりして進んでいく。両者の戦いは，概ね理性の勝利に終わる。ところが希に，理性が抑え切れないほど無意識の感情の力が強くなってしまうことがある。感情の力が強すぎて，理性が抑える暇がないほど急速に感情が膨張して，散財してしまうことがある。世にいう衝動買いというやつだ。そして不思議なことに，そのときの感情が強ければ強いほど，買ったものに飽きるのも早いのである。

これまで何度かそんな目にあった。ある程度は，危ういところで理性が追いついてきてブレーキをかけた，と記憶している。ときにはレジに立っている間に後悔が襲ってくることすらあった。もちろん感情が全力で駆け抜けたこともあったはずなのだが，しかし，何を買ったのかは思い出せない。この部屋の中のどこかにあるはずである。何気なく使っているモノかもしれないし，押し入れの奥で永い眠りに就いているモノかもしれない。ああ，なんと感情とは頼りないものであることか。

衝動買いならまだいい。理由はどうあれモノは買ってしまったのだから，いまさらクヨクヨしてもしかたがない，という一種あきらめのようなもののうちに紛れ込ませてしまうこともできる。やがては後悔も減衰していき，ついには消え去ることだろう。やっかいなのは，感情が日に日に大きく育っていく場合である。十分な力をもった理性が感情を引き戻すのだが，次の日には再び感情が勢いを取り戻している。しかも，どーんと大きくはならないで，理性が引き戻せる範囲にとどまっているというのが癪に触る。最悪なのは，身近なところにプロンプターがいる場合だ。

「今度新しいマシンが出るんです。CPUが変わってクロックもアップするそうですよ」

そうか，あのゲームもずっとスムーズに動くかな。ほかにも触ってみたいアプリケーションもあるしな，などと考えていると，「しかもハードディスク内蔵モデルもあって，価格も抑えてあるんですよ」と追い打ちをかける。明確な反応をしないでいると，1カ月もしないうちに再び，「発売されましたよ。実はいま，家にあるんですよ。いいですよ～」と囁きはじめる。

「今日，見にきませんか」といわれれば，いや，とはいえまい。冬の間になんとか氷づけにして，心の片隅に放り込んでおいた感情に，再び火が点るのが自分でもわかる。なんてことだ。こいつを葬り去ることは，できないかもしれない。そんな予感に戦(おのの)いている。

パロディウスだ！ ―神話からお笑いへ―

Tan Akihiko 丹 明彦

この「パロディウスだ！」は，全編ギャグで埋め尽くされたシューティングゲーム。内容的にも，ほどよいゲームバランスとさまざまな仕掛けで完成度が非常に高い。昨年度のOh!Xゲーム大賞を受賞したのはダテじゃないぞ。

X68000用 5"2HD版2枚組 9,800円(税別)／コナミ ☎03(3264)5678

シューティングゲームとはなんだろう。襲いくる敵の大群，圧倒的な火力。単身飛び込む主人公。厖大な量の燃料と弾丸を消費し，何機もの戦闘機をスクラップにしながら，主人公は死地を突き進む。いつかは負けるとわかっていても。

シューティングゲーム。それは破壊がテーマであるが，何も壊れず誰も死なない，計算機の中の戦闘である。プレイヤーは，そこに敵がいるから撃つ。ただそれだけを行動原理とし，敵を破壊する一瞬の快感を求め，その快感が少しでも長く持続するように生き延びる道を探る。その姿は一種求道者のようでもある。

……なかなか大上段な序文であるが，この解釈は，シューティングゲーム一般に対する，冷静に考えると結構とんでもないシチュエーションを言い表したものである。そして，この解釈は「パロディウスだ！」にはいかにも似合わない。「パロディウスだ！」はスクロールシューティングゲームとしてはまったく異質の存在なのである。

特殊武器でグッドタイミングのセリフ

いかにも全爆という感じ。影がシュール

「パロディウスだ！」は，「横シュー」なんて“通”な言葉を繰り出す，シューティングゲームの達人だけのものでない親しみやすさと，達人をも満足させる内容とを併せもっている。クールでパーフェクトなスクロールシューティングゲームなのだ。

「パロディウスだ！」は誰のもの

「パロディウスだ！」はシューティングゲームとしてはめずらしく，万人向きであるという意見が多いようだ。適度なおふざけ，お色気，かわいいキャラクター，爆発的なセンス，素敵な効果音とBGM，いかにも楽しそうだ。シューティングゲームに殺伐としたものを感じとって敬遠する人も，「パロディウスだ！」には拒絶反応を示さないのではあるまいか。

とはいえ，「パロディウスだ！」はやはりシューティングゲームファンのものである。マニアほどでなくとも楽しめるが，万人向けと言い切るのは乱暴だろう。

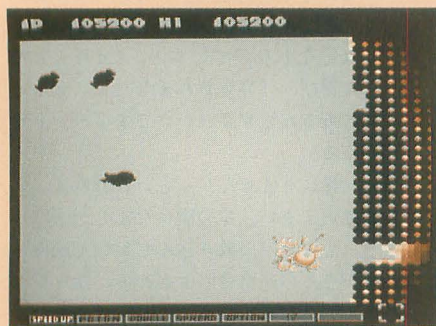
理由その1。パロディを面白がるにはある程度の元ネタを知っている必要がある。「パロディウスだ！」を好きな人は，必ずしもシューティングゲームファンとは一致しないと思うが，「パロディウスだ！」を面白がれるのは，ある意味ではシューティングゲームファンだけであるともいえる。しかし僕のように，あとから「グラディウスII」のカニを見て，逆に大爆笑するようなやつもいるだろう。

理由その2。パロディの要素をとっぱらってゲームの本質部分を透かして見れば，「パロディウスだ！」は結構な難度をもった「グラディウス」系のハードなシューティングゲームだ。スクロールシューティングゲームに詳しくなく上手でもない僕は，このゲームは難度を低く設定しても，1周するのがやっとである。

撃ち込まれると情けない顔になる巨大戦艦

「パロディウスだ！」を気に入ったわけ

僕はもともとシューティングゲームに拒絶反応はないのだが，好んで遊ぶほどではないし，なにより最近の難しけりゃいいといわんばかりの風潮に嫌気がさして，このところアーケードアクション一般から遠ざかっていたのである。正統派シューティングはどれもこれもメカかバイオ生物のオンパレードで，同じように見える。

「パロディウスだ！」は違った。その筆舌に尽くしがたい雰囲気からして違った。ゲームとしてどうこうという以前に，そもそもそのゲーム世界に浸ることが楽しいのである。X68000版の発売前，なにやら怪しげな広告を見かけたが，そのときは気にも留めなかった。「これ買う！」と思ったのは，実物を見たときである。「パロディウスだ！」は，とっくの昔に袋小路に入り込んでいたと思い込んでいたシューティングゲームに，とんでもない突破口を切り開いたように思

ったものだ。
まあ，ゲームを買うのに動機はなんでもいいのである。「わあ，かわいい」でも「なかなかできるじゃないか」でも，「これだ！」というものがあれば。とにかく僕はこれを買った。
で，遊んでみるとわかるように，「パロディウスだ！」は単なるイロモノではなく，シューティングゲームとしてもちゃんと成立している。それどころか超一級品のシューティングゲームである。「グラディウス」シリーズ譲りの優れたゲームシステム，緻密に組み立てられた面，ほどほどの難度，攻略の楽しみ。「ツインビー」シリーズから継承したベルの醸し出す絶妙なゲームバランス。これらが「パロディウスだ！」を末長く遊べる奥の深いシューティングゲームにしている。

タコツボがバリアっていうのもおかしい

涙を流す麗人がボスキャラとは……

面白さの定義は人によって違う

たとえば映画。個人的にはギャグ満載のコメディーものより，ユーモアのセンスが入ったアクションものが好き。「面白いんだぞ，さあ笑え！」といわんばかりの押しつけがましいギャグよりも，無視されることを覚悟しつつ出された，思わず「にやっ」とするようなユーモアの入った演出が好きなのだ。さりげないパロディや皮肉も好物である。本筋からちょっと離れたところで気のきいたことをされると，かなわない。そういう作品と出合うと得した気分になれると思いません？
「パロディウスだ！」は，一見したところギャグの連発に見えるが，「しれっ」と出されたパロディや，ゲームの本筋とは関係ないところで開発者が遊んだ跡が見られ，そういうところが余裕というか，プレイしていて気持ちのよい環境を作り出しているように思うのである。

おわりに

誰でも思いつくようなものは見たってしようがない。自分と違う考え，自分が100年かかっても絶対にたどりつけない考えを聞くのはいいものだ。オリジナルな意見を聞くのは面白い。文章でも，プログラミングでも，そしてゲームデザインでも。
結局のところ，「パロディウスだ！」を気に入ったのは，そのセンスが僕の頭ではどう頑張っても到達できない境地にあるからだ。このような「1本取られた」感覚は，気持ちのよい悔しさであり，負けを認めつつもどこかうれしい。
この感覚を味わう相手の多くは，既存のやり方では分類しきれないような，まったく新しいタイプのゲームで，おおむね海外からやってくる。対して「パロディウスだ！」は，ごくありふれたシューティングゲームというジャンルの中で「違い」を感じさせたという点で偉大だと思う。
それをわかる感性はあっても，それを生み出す才能は与えられなかった……なんて，「アマデウス」の主人公サリエリのような科白のひとつも吐きたくなるような作品に，もっともっと出合いたいのである。

モチベーション（どんなゲームを続ける気になるか？）

僕は，あまりシューティングゲーム，とりわけスクロールシューティングは遊ばない。というよりも，初代X68000の「グラディウス」をクリアして(いまでは我ながら信じられないが，連射なしのスティックでクリアしたこともあったのだ，しかもノーコンティニューで)以来，なんとなくシューティングに飽きてしまったような気がして，その後の加速度的な難度上昇についていけなかったのだ。
算数のドリルが苦痛でしかなかった僕は，同様のものをただ繰り返すことができない。常に目新しいものを求める。そのせいか，僕の好みのゲームは，操作そのものが変わっていて，かつ楽しいものにかぎられてきた。「ポピュラス」「マーブル・マッドネス」「プリンス・オブ・ペルシャ」「レミングス」……。
これらのゲームに共通しているのは，難度の上がり方が教育的であるということである。最初のほうは，とりあえず操作を覚えないことには話にならないので，クリアが容易。そのあと，しばらく進むと壁にぶつかり，そこを突破するためにがむしゃらに練習し，また上手になって先に進む。もちろん，それだけでは好みのゲームとはならないのだが，なぜか教育的であることは共通している。
こうしたゲームの対極にあるのが，硬派シューティングゲームである。もうちょっといっちゃえば「イメージファイト」系である。恥ずかしい話だが，僕は30秒でメゲてしまった。まあ密度が高いというか，ものすごくよくできているだけに歯が立たない。気づいたら死んでいた，というパターン。えらく速いのもつらい。続ける気力が萎えるのだ。
こうしてみると，「グラディウス」系はもう少し丸い。初見でもとりあえず先に進める。強い敵への対応が遅れ，押しまくられてみっともないプレイになっても，どうにか生き残れる。少なくとも数秒間は抵抗できる。ミスしても，対策を考える気になる。どっちがいいとか悪いとかいうのではない，もちろん。
その差は本当に微妙な差である。いまの僕のレベルに依存しているというのも厄介。崖は登れなくても階段は昇れる。「ちょっとだけ難しい」のがいいのだ。それがシューティングゲームをプレイし続けるモチベーションとなる。
話は変わるけど，X68000版「マーブル・マッドネス」の最大の欠点は，クリアが難しすぎること。「very easy」ならよっぽどのミスをしないかぎりクリアできるくらいのバランスがむしろいいのだ。エンディングを見たくてやってるわけじゃない。タイムアタックこそが熱いのだ。難所を鮮やかに抜け，スーパーラップを叩き出す。そういう遊び方だってありじゃないか。ゲームはとてもよくできていただけに，いっそう惜しまれる。あの動きの滑らかさはAMIGA版の比じゃない。
「マーブル・マッドネス」の難しさは，僕からタイムアタックへのモチベーションを削りとってしまったと思うのだ。

シムシティー

Yaegaki Nachi 八重垣 那智

シューティングゲーム研究家の八重垣氏は，実は「シムシティー」にハマっていた。たしかに，シューティングばかりでは疲れてしまうかもしれない。破壊ではなく，創造するという正反対のベクトルにも惹かれたのだろう。

X68000用　5"2HD版　9,800円(税別)／イマジニア　☎03(3343)8911

ゲームを食にたとえると，私はかなりの偏食になるらしい。ゲームセンターでもシューティングゲームばかり遊んでいるし，持っているPCエンジンのソフトなんかも9割がシューティングゲームである。ゲームレビューの仕事なんかに至っては，ほぼ100%がシューティングゲームの紹介である。本人の意思だけでなく，どうやら外見がそれらしく見えるらしいのだが，それの意味するところは甚だ謎である。

とはいえ，好きだからといって本当にそんなのばかり遊んでいるのではない。ジャンルを問わず，実際に面白いかどうかはやってみなければわからないので，極力食わず嫌いはしないようにしている。究極の味とか，至高の味とか気にせずに，とりあえずなんでも食べたほうが，いいゲームに巡り会えると思うのである。

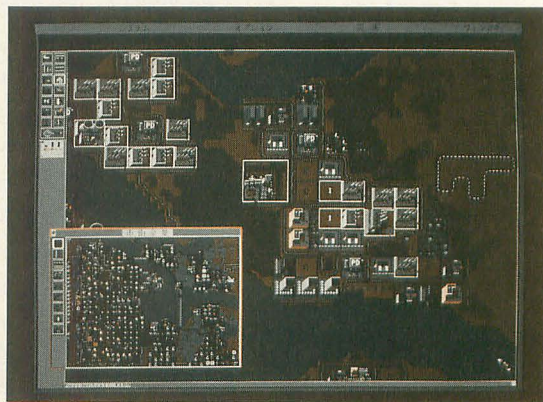
ゲームでないゲーム

シューティングゲームのいいところは，その目的意識の明快さである。一部の邪道なヤツを除けば，ひたすらボタンを叩いて弾を撃っていればいい。たくさん弾を出せばどんどん敵を倒せるし，その行動を継続させるために敵の攻撃をよければイイ。実に簡単で，実に論理的である。そこが，シューティングゲームの最大の魅力である。

しかし，ふと気がつくと，そういったコテコテの目標や目的に突進するゲームとは違うものがほしくなる。とりあえず，反応が速くて，しかもアクセクしないゲームがほしくなる。さらに，目標とか目的とは無縁の遊びができるゲーム，というように制約を加えていくと，おのずとふだん遊んでいるゲームのスタイルから離れていくのは，自然な流れだろう。

これだけワガママな要求を受け入れてくれるとなると，並のゲームでは役不足だ。そこで，これらの条件で(河野景子風に)ブロッキングしてみると，「シムシティー」というゲームが，答えとして選択される。そう，よりによって「シムシティー」なのである。

「シムシティー」というゲームについて，あまり多くを述べる必要はないだろう。ひと口にいってしまえば，都市開発をテーマにしているシミュレーションゲームである。土地の用途を限定し，電力と交通網を整備して町を発展させていくのが，いちおうの方向性である。汚染や犯罪といったものにも配慮するあたりが，原作がアメリカ人によるものであることを匂わせている。

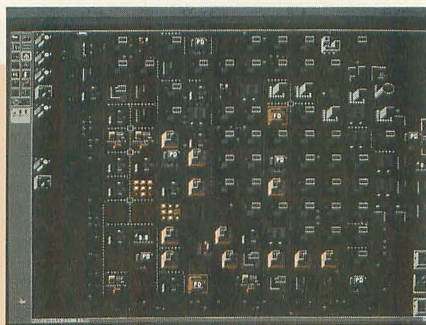
しかし，シミュレーションとはいっても，極めて特殊な部類に入る。シミュレーションというのはそもそも現実の行動を模擬化し，実験するためのものである。数値や理論がふんだんに導入され，できるかぎりリアルに現実に近づけていく。しかし，「シムシティー」は違う。これ以上リアルにもならないし，これ以上デフォルメすることもできないのである。これこそがこの「シムシティー」の特殊性を示す要素以外の何物でもないだろう。

工業地帯は汚染のもと。警察と一緒に隔離する

そして，このゲームには制約も目標もない。目標を無視して遊べるのではなく，そもそもお仕着せの目標などないのである。いちおうシナリオモードには，都市の問題を解決するというような目標が掲げてあるが，そこでも勝手に無視して好き放題やって，実験することができる。荒れ地に100万人都市を作るのも，怪獣を暴れさせるのも(あくまでも)実験であり，誰かに止められたりすることはないのである。

どんな行動をとっても，シューティングゲームのゲームオーバーのように，ゲームから拒絶されたりすることはない。つまり，目標や目的への努力行為に対する快感という，ゲームのシステムはどこにも見えてこないのである。

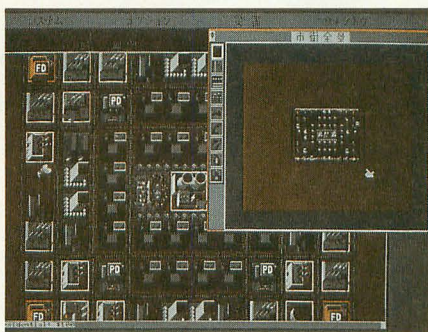
高度な遊び道具

そういったことを考えると，これはゲームではないのかもしれない。しかし，目標を作ることは自由であり，そこまで自分で好き勝手にできるのだから，自分でルールを構築できるひとつ上のゲームと考えるこ

教会や病院は即破壊。専用の土地へと移転

金にまかせて独立国(?)を作ってみたい

ともできるのではないだろうか？　ゲームというよりは，遊び道具といったほうが的を射ているかもしれない。ここまで考えると，シューティングゲームのような目的至上的ゲームのアンチテーゼとして，「シムシティー」の存在を解釈することができるのではないだろうか。

まあ，ここまで強引に理由をこじつけるまでもなく，「シムシティー」をプレイすればそれなりに楽しい時間を過ごすことができる。住宅地を置けばそこには人が住み，電力が不足すれば停電が起きる。そういった反応を観察することに快感を見いだせば，変化していく町を見ているだけで楽しめるのである。それだけ，「シムシティー」は懐が深いということになる。

つまり，あくまでも自主的に目標をもって遊んでいるとはいえ，それは「シムシティー」の中に用意された数多くの選択のひとつでしかない。破壊行為を行って，悲劇的な結果を得ることですら，あくまでも決められたことなのである。が，それはゲームでの安心感につながるものであり，根本において「シムシティー」の論理や因果関係が確固たるものであることを，プレイヤーが無意識のうちに会得する材料になるのである。

そこまで考えれば，「シムシティー」というのは，プレイヤーの頭の中の都市に対する常識をシミュレートしている，ということもできるだろう。現実を直接シミュレートしているのではなく，現実を解釈したモデルや論理をシミュレートしているのである。そういう視点から見れば，最初に挙げた「シムシティー」の特殊性も，説明できるだろう。ゆえに「シムシティー」は，リアルに見えないのに，リアルさを感じることができるのである。

ひたすら対称に町を築くと，こうなる

変なオプションでPC-9801みたいにもなる

何をするのか何もしないのか

結局，このようになんだかんだというまでもなく，思い出したように「シムシティー」に没頭してしまうことがあるのは事実である。次の日の朝早くに用事があるのに徹夜して，「いま寝たら危ない」状態とかには，決まって“somewhere.cty”という町が無秩序に開発されている。時間まで成長させ，保存もしない。そんなことができるのは「シムシティー」だけである。ゆえに「シムシティー」を遊ぶのである。

ガイドブックなどを読めば効率のいい成長のさせ方や，住宅地を重ねて密度を上げるといったテクニックを得ることもできるだろう。通常のように，ゲームとして攻略することも難しくない。しかし，そういう思考から離れ，漫然と理由もなく，絵を描くように町を作っているからこそ，目標や目的ばかりのゲームに追われる自分のスキマが埋められるのかもしれない。

もちろん，/Hオプションを忘れずに(要24kHzモード)。

コンピュータゲームはなくなるの？

ゲームセンターにおいては，「インベーダー」の頃から「テトリス」までは，ゲームはたいていひとりで遊ぶものだった。自分ひとりで，よくいえばストイックに，悪くいえばネクラに遊ぶのが，ゲームプレイヤーの姿であった。

それがいつの頃からか，2人同時プレイのシューティングゲーム，マルチプレイヤーのクイズゲームといった，数人で一緒にプレイするタイプが広がりはじめ，「ストリートファイター」タイプの対戦型格闘ゲームが蔓延するに至っては，ひとりで遊ぶゲームはゲームセンターから姿を消してしまった。パズルゲームですら，対戦の要素は不可欠になり，そもそもそういった，本来ひとり用であったゲームのジャンル自体が，衰退してしまったのである。

カーレースゲームでもそれは同じで，「ファイナルラップ」あたりから，どのカーレースゲームも通信競争型になり，最新の「Virtua Racing」においても，それは踏襲されている。こういった多人数参加型のゲームが，近い将来にX68000へ移植されてくることを考えてみると，どういうゲームスタイルになるのだろうか。それを考えてみよう。

複数のプレイヤー（できたら見知らぬ実力者）と切磋琢磨することによって成り立つような，環境に依存したゲームがX68000に移植されても，とても魅力的だとは思えない。たしかにみんなで騒ぎながらのプレイは楽しいが，ゲーム自体以外に対戦用のプレイヤーが必要になることを忘れてはいけない。プレイヤーの都合がつかないかぎり，その楽しさは味わえず，たとえ家族とプレイするとしても，自分の都合だけでゲームをすることができなくなってしまうのである。

ゲームセンターの場合は，ゲームを求めて足を運べば，そこには同様の目的をもったプレイヤーがいるので，ゲームのある場所にプレイを楽しくする環境が整っているわけである。しかし，自分の家ではそうはいかない。そういうままで考えに入れることの必要がなかった要素がゲームに求められることで，家庭におけるゲームのスタイルが激変する可能性が出てきたわけである。

そんなゲームばかりになってしまうと，ゲームが時と場所を選ぶようになり，限られた条件でしか遊べなくなってしまう。そして，そういった限定された場合にしか遊べないようなゲームを買うということになると，ますます条件が厳しくなってしまうのではないだろうか。単純に考えても，2人でないと遊べないならば，どちらかが持っていればすむわけである。これでは遊ぶ人が多くても，ゲームは少なくてすむということになる。

ゲームセンターでは，限られた機械で多くの人が遊べればいいのだが，それが家庭に入ってきた場合は，必ずしもそれがいいこととはいえないのではないだろうか。そういった意味から考えれば，ゲーム全体が固定された流行に流されてしまうのは，あまり好ましくないということができる。

たしかに，複数のプレイヤーがいないと遊べない，といったようなゲームはほとんど存在しない。しかし，自分の好きなときに好きなように遊べることが，ゲームの最大の特徴ではないだろうか。つまり，複数のプレイヤーがいるときの魅力があるのはかまわないが，それがゲームの本質になってしまうと困るということである。人がゲームに振り回されて，思うように遊ぶことすらできなくなったとき，それはゲームがゲームでなくなるときなのかもしれない。

NEW 3D GOLF SIMULATION

遥かなるオーガスタ

Copyright 1989, 1991 by T&E SOFT Inc.
Made in Japan

Kageyama Hiroaki 影山 裕昭

海外のゲームに影響を受けた日本のゲームは数あれど，逆に影響を与えた日本のゲームはあまり目立たない。しかし，海外のゴルフゲームには，このゲームの影響を受けたとしか思えないものが多いのだ。

X68000用 3.5/5" 2HD版3枚組 12,800円(税別)／ティーアンドイーソフト ☎052(773)7770

眼前にオーガスタ・ナショナルコースの緑が広がる。ここに来たのは2年ぶりだが，青い空と小鳥のさえずりは以前と変わることなく，私を迎え入れてくれた。

なぜ3Dか?

「遥かなるオーガスタ」は3D表示システム“POLYSYS”を搭載した，プレイヤーズアイでコースを回るゴルフゲームである。

8ビット，8色同時表示のパソコンが主流だった頃，ハドソンは「ゴルフ狂」というワイヤフレームで3D表示するゴルフゲームを，そして，ティーアンドイーソフトは「遥かなるオーガスタ」の前身である「3Dゴルフシミュレーション」を発売していた。わりと昔から，ゴルフゲームの3D化は行われていたわけである。

弾道計算を3Dで行うことにより，リアルなボールの軌跡を表示。ディスプレイという2次元の平面の中で，空間の広さ，奥行きを感じさせてくれる3D表示は，広大なゴルフコースを舞台とするゲームにとって，最適の表現手段といえるだろう。

グリーンへのアプローチ。旗つつみか？

ティショット。ボールがぐんぐん飛んでいく

というわけで，3Dゴルフゲーム自体は8ビット機時代からあったわけだが，それらになかったものが「コースの起伏」の表現である。実際のゴルフコースをよく見てみると，たとえフェアウェイといえどもコースはなだらかな曲面を描いていて，まったく平坦なコースなどは存在しない。底が浅くて出しやすいバンカーもあれば，とんでもなく底の深いバンカーもある。「コースのメリハリ」とでもいうのだろうか，そういうものがまったくなかったわけだ。当時の技術力があれば，起伏の再現は決してできないことでもなかったと思うのだが，8ビットマシンのメモリの壁と，処理能力の限界があったのだろう。

話を「遥かなるオーガスタ」に戻そう。このゲームでは，きっちりとコースに起伏がつけられている。コースに起伏がつくと，何が変わるか。ボールが落下した地点が傾斜していれば，当然バウンドも変化する。グリーンにもアンジュレーションがつき，リアルな画像を表示することができる。例に挙げたバンカーも，起伏をつけることで深さが表現される。

ちょっと前に，3D表示は平面上で奥行きの深さ，空間の広がりを表現するのに最適であると書いたことがある。それ以上に，3Dが我々にもたらす恩恵がある。フライトシミュレータなどのように，物体の移動量などのすべてを運動力学や航空力学などの計算式で求めることで，画面上にリアルな動きが再現されることだ。ただ，その代償として，複雑な計算式を解くために処理時間がかかり，描画速度が遅くなることはさけられない。

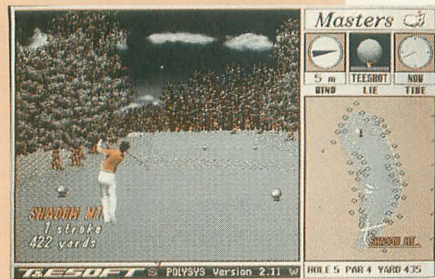
POLYSYSではこれを踏まえて，計算による3D表示に加えて，疑似3D表示をも統括して管理するシステムになっている。具体的にはコースはポリゴン表示にして，木などは何種類かの大きさのパターンを用意

グリーンも凸凹がついていて，読むのが困難

しておき，遠くにあるものは小さいパターン，近くにあるものは大きいパターンを表示している。いらつかないレベルで画面書き換えができるのも，そういった工夫がなされているからだ。

3Dだから

3Dのゲームといえば，「スターウォーズ」や本誌付録ディスクに収録された「SION」を思い浮かべる人が多いと思う。「スターウォーズ」ではコックピットを抜け出て，外部から自機の飛行を眺めることができた。視点を自由に変えて遊ぶことができるのである（「SION」では，座標系をひとつしかもたないMAGICの制限で，残念ながら実現できなかった）。

この点については，「遥かなるオーガスタ」の対応はいまいち。これだけよくできたコースデータが，ショット位置を中心とした360度の範囲でしか見ることができないのは非常にもったいない。もっといろんなことができたはずだ。上空を飛ぶボールから見た視点，というのも面白いだろう。コース脇に何台かテレビカメラを設けて，そこから見たボールの行方なんかも表示できたら面白かったと思う。さらにはホールインワンや，「これは」と思ったショットを保存できるようにして，あとでリプレイできるようにしておくと，やり込めばやり込むほど見応えのあるデータが増えていく

当然ながら，木には当たり判定がある

こういうときはピンに当てるつもりで

ので気合も入るだろう。

さて，3Dは計算に時間がかかり，表示速度が遅くなる。が，それは10MHzのX68000での話。16MHzのXVIで遊んだときは，体感速度が2倍ではないかというくらい快適になった。そして，俄然ここで注目されるのがX68030である。結論からいうと，X68030で「遥かなるオーガスタ」は動きます。もちろん，ちゃんと速くなる。

3Dではないが

さて，ここまでは3Dの魅力と，「遥かなるオーガスタ」における3Dシステムについて書かせてもらった。いちおう，ここで3Dシステム以外のところについても少しいわせてもらおう。

ゴルフを3Dで表現するという部分だけ見れば，技術的にはすでにほかのソフトウェアメーカーの追随を許さない領域まで達している。しかし，よくできているだけに，それ以外の部分で気になる点がいくつかある。特にX68000版ではディスクアクセスの長さが目につく。計ったことはないが，電源オンから1番ホールのティーショットに入るまでに，おそらく3～5分かかるだろう。待ち時間のほとんどがディスクアクセスというのはいただけないが，逆にハードディスクなどの高速なデバイスにデータを置くだけで改善できる問題でもある。

マニュアルプロテクトでバックアップの作成が簡単にできて，ハードディスクへのインストール可能。コマンドラインで起動，終了後はOSに復帰。これはIBM PCやMacintoshのソフトでは当たり前となりつつあるシステムであるが，X68000では当たり前とはいえないようだ。特に，ディスク枚数やアクセス回数が多くて，インストールできて当たり前というゲームで実現されていないことがよくあるのは少し残念。

*　　　＊　　　＊

原稿を書いている最中に初めて知ったのだが，ボールから見た視点というのは，某国民機で発売されている「ペブルビーチの波濤」ではすでに実現されているらしい。しかも，ハードディスク対応や，視野の拡大などもサポートされているようだ。う，うらやましいぞお。

思い出のゲーム（または，自己回顧録）

Oh!X編集室に出入りするようになった18，9歳の頃，私は渋谷にある専門学校へ通う学生だった。よく自主休講して，「グラディウスII」「アフターバーナーII」「R-TYPE」などのシューティングゲームをゲーセンにプレイしにいったもんだ。思い起こすとOh!Xのスタッフになったのも，いくつかのゲームとの運命的な出会いがあったからである。

シューティングゲームを好むようになったのは，「アルフォス」をプレイしたときからだろうか。その「ゼビウス」に似た縦スクロールシューティングゲームは，いまは分解されて押し入れで永眠しているPC-8801mkIIで動いていた。ハードウェアスクロールのないPC-8801で滑らかなスクロールを実現していたのは印象深い。ゲームをテンキーで遊ぶのが当たり前の時代でもあった。おかげでダンプリストの入力だったら，誰にも負けないくらいに速くなった。

高1の秋に，本誌のライターで友人でもあるS.K.氏がX1turboを買った。シューティングゲームが大好きだった私は，“これで，あのX1版「ゼビウス」がただで遊べる”と心の底から喜んだものだ。このときX1turboと出合わなかったら，いまの私はなかったといえる。それほど，S.K.氏とX1版「ゼビウス」の与えてくれた影響は大きかった。X1シリーズのPCG機能がうらやましかった。

そして，翌年の冬にPC-8801SRが発売された。発売されるゲームもSR専用版ばかり。くやしくなった私はSR専用の「グラディウス」を買ってきて，メインプログラムにパッチを当て，PC-8801mkIIで無理やり動かした。SRがハードでやっていることをソフトで行うプログラムを作り，ディスクエディタでディスクの空きセクタに書き込んだのだ。なんか哀れな行為ではあるが，こういう涙ぐましい努力を重ねて，Z80のアセンブラを学んでいったことが，S-OSをPC-8801へ移植するための予備知識となった。

S-OSの移植がきっかけで，Oh!Xにお世話になるようになった。初めて2ページ書かせてもらったゲームレビューは，「ツインビー」（“出たな”じゃないよ）であった。その頃は自分でX68000を買う金もなく，編集室のマシンで遊んでレビューを書いていたんだっけ。難しくて2周目にもいけずに，自分はたいしてゲームがうまくないことに気づいた。そして，山田君や横内君のゲームの腕を見て，世の中にはときどきとんでもないヤツがいるものだ，と痛感した。

自分で買ったソフトのなかでは，「ポピュラス」がいちばんのお気に入り。もちろん，500面を制覇した。ほかにはX1turboで遊んだ「スタークルーザー」とPC-8801mkIIで遊んだ「ウィザードリィ」が印象に残っている。「スタークルーザー」は，ポリゴンの戦闘機が画面を飛び回るデモが脳裏に焼きついている。「ウィザードリィ」はこつこつとマッピングした苦労と，ムラマサブレードを見つけたときの喜びが忘れられない。もちろん，私がX68000に移植したMAGICを使ってくれた「SION」シリーズも，大好きなゲームだ。

ねじ式

Nakano Shuichi 中野 修一

これを単にアドベンチャーゲームと呼んでいいのかどうか。いや，ゲームと呼ぶことさえ，はばかれる。文章，音，そして，絵が見事に統一されたその世界においては，謎解きなど付属物にすぎないのだ。

X68000用 5″2HD版3枚組 12,800円(税別)／ツァイト ☎03(3299)0461

ねじ式とは

ねじ式は，
つげ義春という，
ひとりの作家の世界をもとに，
構成された，
小さな「冒険」である。

＊　＊　＊

「ねじ式」を楽しむために，「つげ義春」という作家を知っている必要はない。なぜなら，これはつげ義春が描いた「ねじ式」の世界ではなく，つげ義春の世界をひとりのゲームデザイナーが独自に再構成した作品だからである。それが誰かは私は知らない。単につげ義春の作品からキャラクターを抜き出しただけでは，絶対にこの作品は成立しない。誰かがつげ義春を素材に「つげ義春の世界」というものを創り出しているのだ。だから，プレイヤーはその中に入っていくだけでよい。そこにすべてがあるのだから，特別な予備知識は必要ない。

ゲームはひとりの老作家Tの自伝小説執筆を巡っての，虚構と追憶の混ざりあった世界に展開される。欠落した記憶と秘められた自らの過去，その郷愁の中へと旅に出る……。

しかし，実際，こういった設定にどの程度の意味があるのか？　プレイヤーはあまり気にもとめないだろう。それはこのゲーム内での現実であり，つげ義春の作品から最もかけ離れた部分だからだ。

プレイヤーはすぐに現実からひとつの過去へ追い立てられる。

ゲンセンカンから

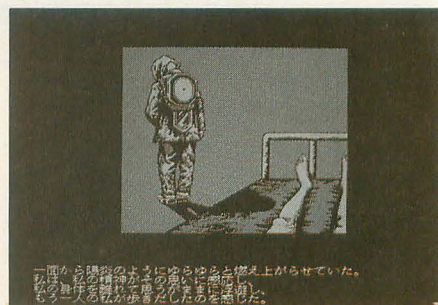
その町はまさに序章である。プレイヤーは試行錯誤とゲームの進行方法，そしてこの世界の空気を味わう。いきなり行き詰まる人も多い。

しかし，この町には人の息づかいはない。謎を提示し，扉を開く。この時点ではまだどんなゲームかなど見当もつかないだろう。しかし，確実に物語は始まっている。

＊　＊　＊

そして，村である。コバヤシチヨジがいる。キクチサヨコがいる。シンデンのマサジがいる。

村はこれらのキャラクターの生活空間である。ここではプレイヤーは旅人にすぎない。彼らの生活は彼らのものであり，プレイヤーの行動が彼らの運命を変えることはない。この世界では悲しいほど異邦人である。以前，泉大介氏がコバヤシチヨジになんとか靴を買ってやろうとした気持ちはよくわかる。

ここには，アドベンチャーゲームでのフラグを立てるためにだけ存在するキャラクターとは，違った人たちが生活している。そして，普通のアドベンチャーゲームで世界の運命を支配するはずのプレイヤーは，あまりに無力である。自己の存在意義に対する疑念，村人へのかぎりない郷愁。

ゲームを開始するとそこはもう別世界

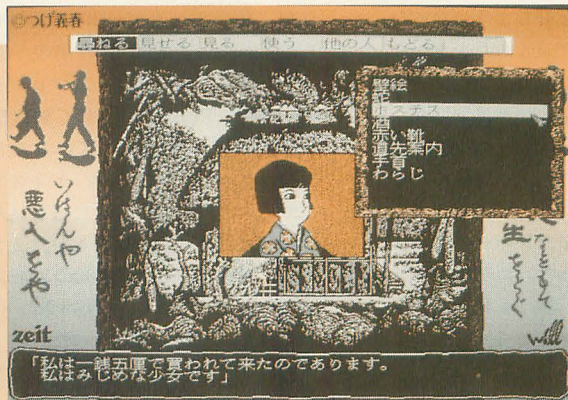
静寂は「ねじ式」での基本的なBGMとなる。展開は遅々として進まず，途方に暮れること，漠然とした不安がゲーム要素に取り入れられていることを知る。そして，森の中で恐怖というものを知る。

ゲームシステム

ここで，ねじ式のゲームシステムを見てみよう。

ゲームの進行には無関係な選択肢。いつフラグが立ったのかもわからない。同じ台詞を繰り返すキャラクター。なんとなく行き詰まることもけっこう多い……。本来なら，「馬鹿野郎」と投げ出してしまうようなシステムである。しかし，多大な魅力を持って人をひきつける。

ほとんど単色のグラフィック，単純なBGM，無論アニメー

ここはまだ本編に入る前の場面

雰囲気に合ったセリフの数々に感動

ションしたりしゃべったりはしない。それでも，いま流行のアドベンチャーゲームの10倍は面白い。そういえば，昔，言葉探しのアドベンチャーゲームを夢中になってやっていたのはなぜだったろうか？

それはたぶん，ひとつのお話をなぞっているだけではなく，ひとつの世界の中で思う存分「冒険」ができたからではないだろうか？

＊　＊　＊

森を抜け，
しんしんと降り積もる雪の中……，
郷愁を乗せた列車に乗り，
少年と出会う。
そして。

＊　＊　＊

「ねじ式」の世界はいくつかのシーンで構成される。それぞれがなぜかノスタルジックであり，ゲーム世界に思いきり浸ることができる。遅いゲーム展開など，まったく苦にはならない。いっときでも長くこの世界に浸っていたいとすら思わせるものがある。「ああ，日本人に生まれてよかった……」。これがゲーム中の感想である。

しかし，物語が展開していくと名残は尽きぬものの，その世界を去らねばならなくなるのだ。新たな展開に期待しつつ，次の世界に移行する。

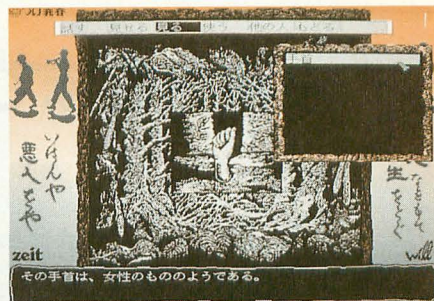
このあたりで少しずつひとつの疑問が頭をもたげてくる。不安なのは，ゲームがどのように展開するのかがまったく読めないことだ。ましてどのようなエンディングを迎えるのかなどは見当もつかない。いったいこの先どうなるんだろうか？

そして異邦人であったプレイヤーがだんだんゲーム中のキャラクターとして行動していくことに気づく。

エンディングへ向けて

どこかにいるはずの作家Tとしての自分，プレイヤーとしての自分，そして現実から遊離したゲーム内の世界，そしてゲーム内の現実世界。これらのどうしようもなくバラバラな要素がしだいに接近し，思わぬかたちで急速に収束していく……。感嘆符を連発し，ひとときも目を離せなくなるクライマックス。このあたりの手腕は凄い。本当に凄い。

こんな作品を作り出せる人がいたということが，たまらなくうれしくなる。こうなると原作の持ち味や世界観は二の次だ。緻密な構成，グラフィック，音楽，そして演出。なにをとっても素晴らしい。コンピュータゲームってのも捨てたもんじゃないと思わせる。

だらだらした文章はなく，制作者の知性を感じさせる文章。ゲームはゲームデザイナーとの対話である。とすると，この人とは話してみても楽しいのだ。

樹海の入口に生えている妙な手首

記録するときもいちいち雰囲気がある

個人的には，PC-9801から移植されたゲームの中で移植されてよかったと感じたのはいまだにこのゲームしかない，と思っている。題材が非常にマイナーなのが惜しいが，「ねじ式」は国産アドベンチャーの最高峰といってもいいだろう。

デザイナーとの対戦

考えてみればアドベンチャーゲームの少ないこと。いや，それ以前にアドベンチャーをやらなくなった。けっこう忙しいので時間がかかるとわかっているものはできないのだ。いちばん最近やったアドベンチャーというのが，「3時間で終わる」と聞いて始めたら2時間で終わったサイメビだから，ずいぶんやってないことになる。ああいうストーリーを追うものだとなぜか作者の意思に反した行動をするらしく，うまく話がつながらないことが多い。ちなみに，現在まわりではなぜかフランス語のアドベンチャーゲームが流行っている。

昨年いちばん遊んだのは，それでもポピュラスIIだった。マウスを長時間握り締めているため，気がついたときには右腕が筋肉痛を起こしたものだった。続けていたら腱鞘炎になったかもしれない。画面に対して集中しているので目も疲れる。で，くたくたになりながら，あの攻撃にはどう切り返すべきかと絶えず戦略を練っている。

対戦用のマップを作ろうといろいろやって，何日もかけて何台ものマシンを使って，結局あきらめた。

いっちゃあなんだがAMIGA版にもバグが多い。前作で通信中によく落ちるのも原作譲りだった。今回も原作からして問題があるんじゃないかと疑っている。世界的なソフトであるにもかかわらず，そのへんは改善されていないし，海外のネットワークでも特に話題になっていない。ふと，対戦ってそんなにマイナーなのだろうかと考え込んでしまう。イコールコンディションでない対戦なんて対戦といえるか？

やはり基本は対戦だ。

国産のパソコンゲームでもっとも優れていると思うのはフラッピーだ。「洗練された難しさ」がいい。限られたフィールド内ではゲームデザイナーとプレイヤーが対等である。これはゲームデザイナーとの戦いだ。

類似品で倉庫番とかキャッスルなどがあったが，ぱっと見て「無限に難しくできる」のと「どこで失敗したかわからない」ので私の中ではクソゲーに分類される。作るのは簡単で解くのは異様に難しいパズルなんて他人にやらせるものじゃない。

イメージファイト。誤解されやすいが，あれは見かけほど反射神経は要求されない。リアルタイムパズルのようなものだ。仕掛けが巧妙で解法は美しく答はひとつじゃあない。パターンを知らなきゃ解けないというのは筋違い。解法は自分でみつけるものだ。力押しできるあいだは反射神経が必要でも，復活パターンはパズルである。必ず解けるとわかっているから何度でも挑戦できる。やはり2周目2面からの復活が好きだ。

こうしてみると，やはりそのゲームを作った人というのを無視して話は進められない。ひとりでやるゲームはそのゲームの作者との対戦ともいえる。やはりゲームは度量の大きい人に作ってもらいたい。

＊　＊　＊

余計なお世話といわれるかもしれないが，ポピュラスIIのフルパワーパスワードを掲載しておく。

ADKIWCKBCNZEZIWX

だ。私は途中で挫折してしまったが，いつかはゼウスを倒したいものじゃないか。では健闘を祈る。

今夜も朝までPOWERFULまあじゃん2

Shiba Mamoru 司馬 護

ジャンルにかかわらず，よくできているゲームは長く遊べる。なかでも，いいテーブルゲームはとても長持ちする。ルールはすでにこなれているから，システムの出来がよければ面白さに直結するということかな。

X68000用　5″2HD版4枚組　7,800円(税別)／デービーソフト　☎011(807)6700

「昔のソフトでよく遊ぶゲームですか？」

好きなゲームを１本選ぶといっても難しい。Oh!Xのライターは個性派揃いだから，九種九牌，もとい，多種多様なゲームが出揃うに違いない。そんななかで，私はこの麻雀ゲームを推薦したい。

雀・ギャバン

X68000にも麻雀のゲームは数多く存在する。新しいところでは,「雀JAKA雀」とか「麻雀遊園地」など，ちょっと古めで「哭きの竜」「麻雀悟空」，そして最古参の部類に入るのが,「麻雀狂時代スペシャル」や「今夜も朝までPOWERFULまあじゃん２」（以下「パワフル麻雀」）である。

他機種ではさらに選択肢が広がり,「スーパーリアル麻雀P3」だとか「ぎゅわんぶらあ自己中心派」「インテリ雀師四人麻雀」「まじゃべんちゃーねぎ麻雀」なんてものもある。

さらにゲームセンターに置いてあるものまで含めると，何種類あるのか見当もつかないほどだ。こうやって考えてみると，いかに多くの麻雀ゲームがこの世に存在しているかに驚かされる。

ちなみに，「インテリ～」の作者は意外と身近な人であることが判明している。FORTHで記述され，“Simple is the best”を地でいくゲームであった。

雀ヌ・ダルク

これだけ多くの麻雀ゲームがあると，それぞれの方向性がまったく違っていても，なんら不思議ではない。ゲームセンターにあるもののほとんどすべては，脱衣もので ある。しかし，パソコン上では脱衣のほかにも，打ち手の個性を狙ったものや，娯楽に徹しているもの，ウケをとりにいったものなどに分けられるだろう。たいていのゲームはいずれかに当てはまるはずである。

では，「パワフル麻雀」はどうだろう。実はどれもが当てはまる。というと欲張っているようだが，ゲームバランスは崩れていない。なぜなら，まったく別のモードとして，４種類の麻雀がフォローされているからである。４人打ちの一般的な“ノーマル麻雀”，ドンジャラのような“ぽこ麻雀”，脱衣ものの“エキサイティング麻雀・お姉さん組”，ストーリー仕立ての“さすらい麻雀”。このような麻雀ゲームは，このソフトをおいてほかにあるまい。

個々のゲームに分けて考えると，もっと面白いゲームや個性的なゲームを挙げることができるかもしれない。しかし，１本のソフトの中に，ここまでいろいろな麻雀が統合されたゲームは貴重な存在である。統合化環境版麻雀と呼んであげよう。

ちゃんとした麻雀がいい

ふと麻雀をしたくなるときがある。それも４人打ちの普通の麻雀がいい。ひっかけ，読み，多分に駆け引きを含んでいる。戦略という言葉が大げさではない，勝負の世界。脳ミソの調子がよすぎて困ってしまうようなときなのに友達が３人集まらないなら，相手はパソコンだ。

「こてんぱんにのしてやる」

負けない自信はある。引きのよさ，読みの深さ，自分でも恐ろしいほどツボにはまる。勝利の女神がベッドでウィンクしているようなものだ。

こんな気分のときは“ノーマル麻雀”。

「ふぅ～」

ちょっとHな麻雀がいい

ふと麻雀をしたくなるときがある。それも女の子を脱がせるやつ。実際に体験できないってところがいい。金曜の夜をひとりで過ごす，ちょっとむしゃくしゃするような日には，こんな不条理の世界が楽しい。

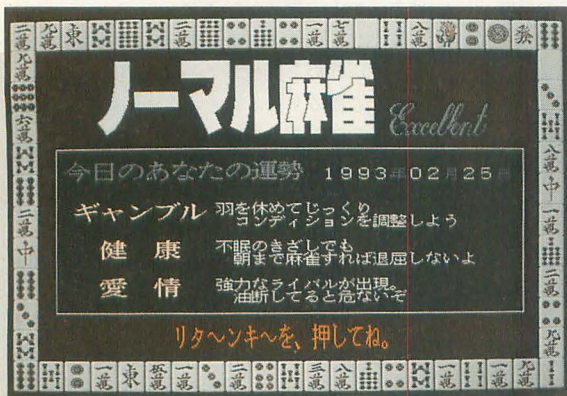
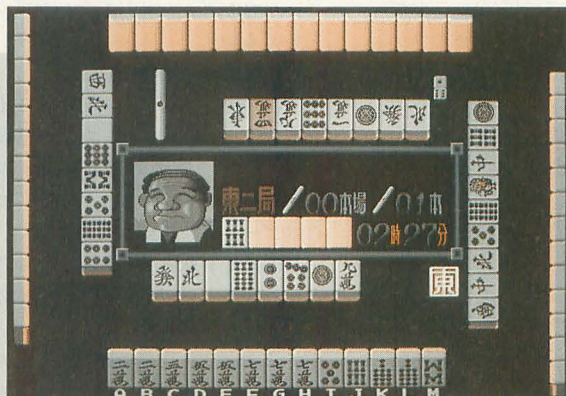
「てこずらせやがって」

局面ごとの勝ち負けは，まんま恋の駆け引きのようだ。積み込んでいるとしか思えないような相手の配牌ですら，かわいく思える。

麻雀をする相手はひとりだけだが，脱いでくれる女の子は８人いる。

脱ぐ女の子はバラエティに富んでいるので，つい目移りしてしまいがち。

こんな気分のときは“エキサイティング麻雀”。

「はぁ～」

生年月日でその日の運勢を占ってくれる

ノーマル麻雀はもちろん４人打ちの本格派

肩の凝らない麻雀がいい

ふと麻雀をしたくなるときがある。それもドンジャラのようなくだらないやつがいい。金を賭けるわけじゃない。起きぬけや寝不足などで，頭がボーっとしているときに軽めの刺激を求める。迎え酒ならぬ，迎え麻雀。

「リーチ」

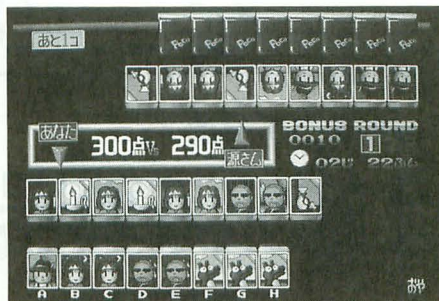
敵がかけるリーチが興奮剤となって，重たいまぶたを押し上げる。頭は回っていない。捨て牌は運試しに近い。

こんな気分のときは“ぽこ麻雀”。牌は同じソフトハウスの「うってぃぽこ」に出てきたキャラクターやお酒など。役の名前も「酒だらけ」「美女と野獣」なんていうのがある。

「ふっふ～ん」

エキサイティング麻雀は好きな女の子と

見た目に楽しいぽこ麻雀

さすらい麻雀は双六と麻雀で世界を回る

こんなだるい気分のときは“さすらい麻雀”でもいい。役満しばりだとか，清一色麻雀，役指定麻雀なんていう，ちょっとほかでは考えられない構成をしている。麻雀とはまるで関係のない，クイズやロシアンルーレットまである。

「ひゅ～」

雀・雀・雀・じゃ～ん

「パワフル麻雀」はこういった本筋だけではなく，細かな点でもアミューズメント性が高いのである。

最初にプレイヤーを登録する際に，誕生日や顔のモンタージュを作成する。誕生日に起動すると，画面には大きなケーキが現れるし，季節によっても画面が変わる。ちなみに顔はノーマル麻雀のときに現れる。あんまり美形に作りすぎると，あとではずかしいのはなぜだろう。

このような小ワザを見せるソフトとしては，ほかに“こんばんはお姉さん”の「ドラゴンスピリット」，“宇宙戦艦ヤマト”の「銀河英雄伝説」などがある。また，「パワフル麻雀」はPC-9801からの移植モノだが，グラフィックは全面的に描き直され，美しいグラフィックを楽しむことができる。しかし，忘れてはならないのは基本のシステムがしっかりしているからこそ，こういった細かい部分がなおさら映えるということである。

冴えわたった，悶々とした，疲れきった脳ミソには，一服の清涼剤とでもいうような「パワフル麻雀」はいかがだろうか。別売で女の子のデータ集が発売されている点も見逃してはならない。

6.15mのハードル

最近のゲームに多い傾向であるが，どうも作り手と受け手の駆け引きが減った気がする。薄々感づいてはいたのだが，確信をもったのは「ウィザードリィIV (RETURN of WERDNA)」を解き終えたときである。

「ウィザードリィIV」には幾重にも用意された周到なワナがある。それは単にフラグセットというものではなく，冒険者を拒みつつも待ち望んでいると感じさせるものである。基本的には迷路に関するものか，アイテムの使い方に関するものしかないのだが，それでも面白い。謎のレベルがだんだん上がっていくという教育的な配慮があるのも，輸入ゲームの特徴か。

ゲーム自体の目的は，失われたアミュレットを取り戻すというシンプルなものである。しかし，その目的以外の情報はほとんど欠如している。そもそも誰がアミュレットを持っているのかすらわかっていない。徘徊しているのはすべて敵なので，聞き込みもできない。勘が鈍い人は最初の部屋から出ることすらできないだろう。あくまでもエキスパートオンリーという前提のゲームなのだが，あまりにも「ないないづくし」である。

そこに御言葉師MRONの登場する余地が生まれる。彼は神託としてヒントを出してくれるのだ。しかし，それも間接的，抽象的なものなので，プレイヤはない知恵をふりしぼらなくてはならない。彼は連想ゲームにおける加藤芳郎のようなものだ。断片的に判明してくるデータをもとに推論を立てて，先に進む努力をするのである。ここに作り手と受け手の駆け引きが存在している。

駆け引きというものはRPGで特に表現しやすいが，実際にはどのジャンルにも存在している。シューティングゲームならば敵の出現パターンや難易度，教育性も駆け引きになる。これらを総称してバランスという言葉で置き換えることもできるだろう。バランスがいいゲーム，それはすなわち面白いゲームである。ユーザーのレベルや成長の度合いを見ながら，さらに上を目指すような上昇志向が，ゲームから感じられなければならないのだ。

突然，6.15mのハードルを飛び越えろといっても無理な話である。たいていの人は何回かチャレンジして己の限界を悟り，あきらめてしまうだろう。しかし，自己ベスト＋1cmを飛んでみろといったらどうだろう。手ごろな目標に対して努力をする人が増えるのではないだろうか。

ちょっと抽象的な話になってしまったが，最近のゲームでは立ち上げる前に気合を入れなければ遊べないものが多い。それはハードルの設定が高すぎるからではないだろうか。肩が凝るゲームだとも表現できよう。この原因を考えてみると，ほかでもなく駆け引きがうまくいってないのだ。受け手が難しすぎると感じていることを作り手はわかっていない。プレイヤーにはレベルがあるのだから，超ムズいレベルがあっても結構。しかし，初心者をないがしろにするようなゲームが多いのはいただけない。

また，レベル調整を人間に求める対戦ゲームを，安易に粗製濫造するのは考えものである。いまや対戦格闘ゲームの代名詞となった感のある「ストリートファイターII」は，ひとりでプレイしても面白い点を忘れてはならない。最近では駆け引きがうまくいっている貴重な例である。

コンピュータアーキテクチャ編

切り替え式加減算器の完成

Misawa Kazuhiko
三沢 和彦

今月で，いよいよ加減算器回路が完成します。基本的に製作回路は2，3月号で製作したものの流用ですが，具体的にどの部分を流用しているか，どのように結び付いているかをしっかり確認しながら製作してください。

先月は加減算器の出力表示部分の製作だけで終わってしまいました。しかし，その部分だけでTTL ICが3個も必要となるので，初心者の皆さんには十分歯応えのある回路だったのではないかと思います。

今月はいよいよメインの加減算器を完成させたいと思います。加減算器の回路図(図1)は大部分が2月号に示したとおりですが，演算結果を表示する部分に関しては3月号の回路に接続する形で設計してあります。とりあえずは3月号の表示回路の部分は一切忘れてしまってかまいません。

注目してもらいたいのは，図1中で点線で囲まれた2月号からの変更部分です。

部品のレイアウト

さっそく実体配線図から入っていきたいところですが，まず部品のレイアウトが重要です。部品のレイアウトを考えるにあたっては，機能に従ってブロック別に考えていくのが鉄則です。部品のハンダ付け側から見たレイアウト図を図2に示します。

スイッチ関係を最下段に並べるとすると，信号の流れとしてスイッチの次にくるのが反転用のLS86なので，LS86を最も近くに配置します。TTL ICは先月の実体配線図にもあるように基板の横1列に3個並ぶようになっているので，スイッチの真上にLS86がくるようにするのがベストです。

しかし，実際問題としてスイッチの周辺には抵抗のスペースが必要なため，スイッチの真上に持ってくることができませんでした。これはあとで実体配線図を見ればわかると思います。結果的に図2のように下から2段目の真ん中に配置します。

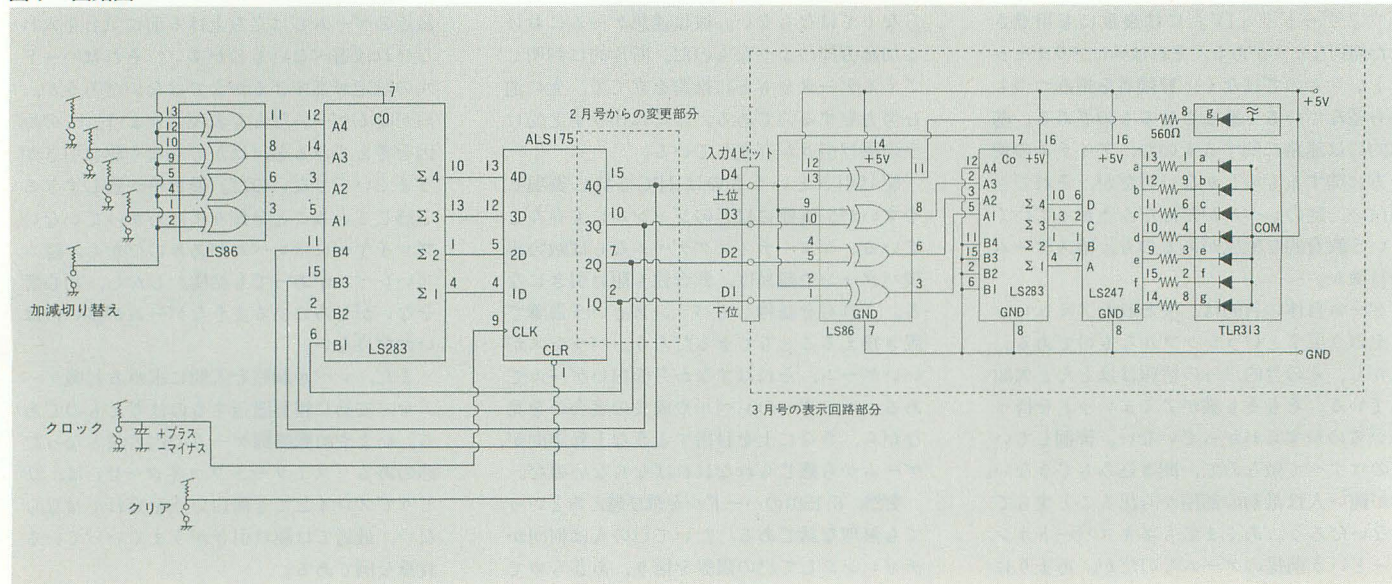
逆に演算結果を出力するレジスタのALS175は，表示回路の近くに持っていく必要があるので，上のほうに配置します。今回は下から3段目の左端に取り付けることにしました。

そして，メインの加算器LS283は，LS86から入力を受け取ってALS175に出力を受け渡すので，必然的にLS86とALS175との間に入ります。これまで述べてきたように，すでにLS86とALS175の位置は決まっており，その間にLS283を配置するとなるとL字型に並ぶことになります。

実際に製作を終えたあと改めて眺めてみると，加減算器に使う3個のTTL ICを縦

表1 部品表

LS86	1個	40円
LS283	1個	70円
ALS175	1個	60円
16ピンICソケット	2個	@35円
14ピンICソケット	1個	30円
スライドスイッチ	5個	@20円
押しボタンスイッチ	2個	@30円
10Ω抵抗	7本	@1円
10μF16V電解コンデンサ	1個	25円
配線材	少々	

図1 回路図

3段に並べたほうがよかったかとも思いますが，もう作ってしまったので，このままの形で進めていくことにします。先月の表示回路は図2に示すように基板の最上段から並べているので，現在のままではある程度のスペースが空いていることになりますが，そこには来月以降に製作する回路が載る予定になっています。とはいえ，今回の加減算器のメインの部分は今月製作するところまでで完成となります。

製作実習

では，実体配線図（図3）を見ながら配線していきましょう。まず，ICソケットをハンダ付けします。いつものように，7番（8番）ピンはGND，14番（16番）ピンは＋5Vなので，内側に折り曲げて，基板の真ん中を通っている2本ラインを一緒にハンダ付けしてしまいます。今回の回路ではそのほかにGND直結あるいは＋5V直結の足がないので，残りのピンはすべてそのままそれぞれのパターンにハンダ付けします。

今度はスイッチ群を取り付けましょう。前回までは2ビットひと組のDIPスイッチを使ってきましたが，今回の回路は4ビットのDIPスイッチを使うことになります。すると，DIPスイッチ自体が小さいためにペンの先のようなとがったものを使わないと，スイッチを切り替えられません。

そこで，今回は1ビットずつ独立したスライドスイッチを使うことにしました。スライドスイッチには図4のように3端子ありますが，真ん中の端子が共通でスイッチをスライドさせた側の端子とそれぞれ導通するようになっているため，特にON/OFFの向きはありません。4ビット入力と加減算の切り替え用に計5個取り付けますが，加減算切り替え用のスイッチだけほかの4個と少し間隔を空けることにします。

押しボタンスイッチはこれまで使ってきたものと同じです。初めから導通している2組の足がそれぞれ横に並び，スイッチを押したときに上下の足が導通するように取り付けてください。

次に，これらのスイッチにつながる抵抗を取り付けます。抵抗の足は，差し込んだら折り曲げて，スイッチの足のところまで延ばし，そこに一緒にハンダ付けします。反対側の足は折り返して＋5Vラインにハンダ付けします。これらのスイッチからTTL ICにつなげるところはすべてジャンパ線になっているので，あとから配線することにします。

また，クロック端子にはチャタリング防止用の電解コンデンサが必要なので，忘れずに取り付けてください。なお，電解コンデンサには極性があるので，コンデンサ本体に付いているマイナスの表示を確認するようにしましょう。

電源端子は先月の7セグメントLED表示回路で取り付けたものが，そのまま基板全体に使えるので，今回新しく製作する必要はありません。

以上で部品の配置は終わりました。あとは部品の間を配線材でつなぐ作業だけです。スイッチ群の片方の端子をGNDに落とすために基板のいちばん下にスズメッキ線をはわせてしまいます。

あらかじめいっておきますが，今月の回

図2　部品のレイアウト

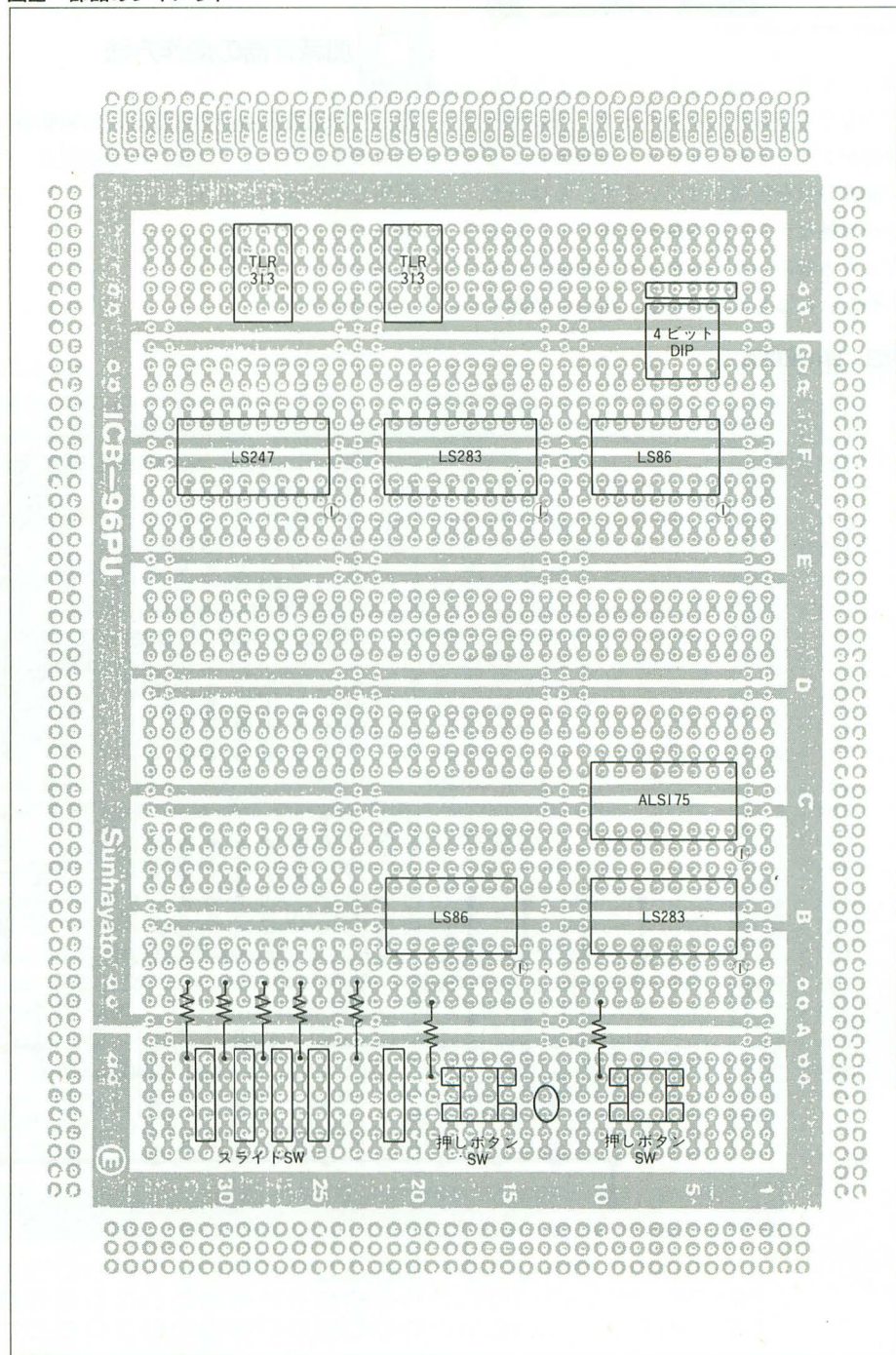
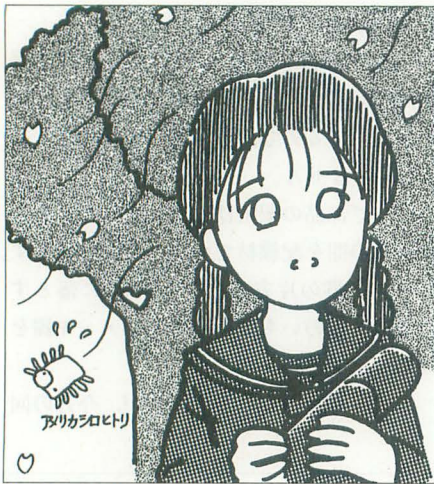



illustration:Y.Kawahara

路は，ビニール被覆線でジャンパを飛ばしてつなぐものがかなりあります。特にデータ線が2ビットから4ビットに増えたことが配線の量を倍増させています。とはいえ，今後複雑なデジタル回路を製作していくとなると多数のパラレルデータの配線をしていかなければならないので，根気も必要となってきます。実体配線図上の対応をよく確認しながら，間違えないようにハンダ付けしていってください。

最後に表示回路への4ビット出力をジャンパ線でつないで完成です。表示回路は前回製作した回路に変更はなく，図5のように配線を加えるだけでOKです。なお，表示回路のチェックに使った4ビットDIPスイッチはすべてOFFにして，LS86への入力がDIPスイッチ側からはHレベルになるようにしておかないと，正常に表示されません。

加減算器の操作方法

以上で10進数表示の切り替え式加減算器が完成しました。まずは操作の手順を追って説明していきましょう。

例として，

2+3=5

2−3=−1

の2つの演算を考えてみます。

●2+3=5の場合

1) クリアボタンを押して表示を0にする

2) 加減算切り替えを加算にする

3) 入力スイッチを2（=0010）に設定する

4) クロックボタンを押して2を入力する(実際は0に2を加算していることになる)

5) 表示が2になる

+6) 加減算切り替えは加算のままにする

+7) 入力スイッチを3（=0011）に設定する

+8) クロックボタンを押し3を加算する

+9) 表示が演算結果の5になる

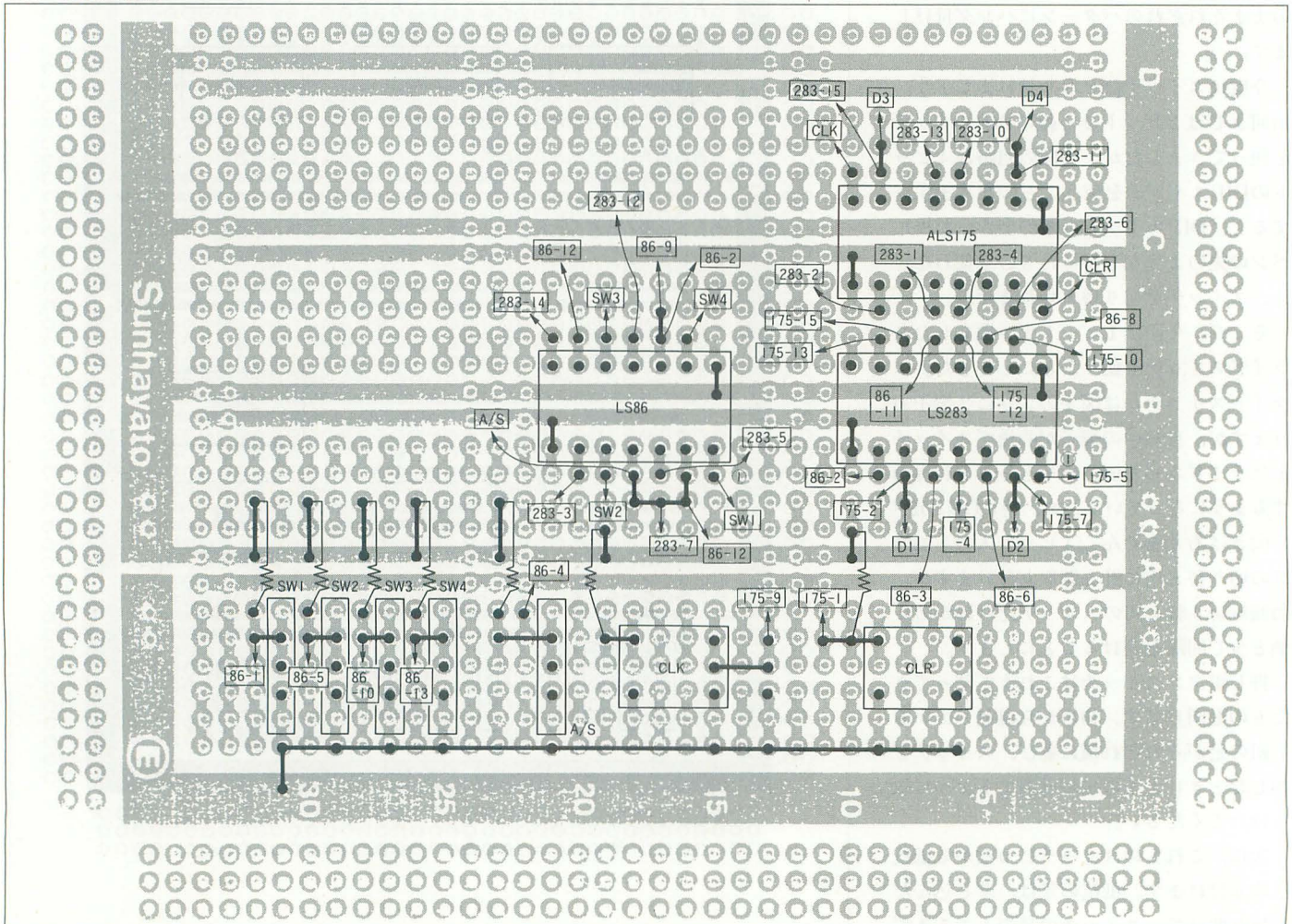
●2−3=−1の場合

1) クリアボタンを押して表示を0にする

2) 加減算切り替えを加算にする

3) 入力スイッチを2（=0010）に設定する

4) クロックボタンを押して2を入力する(実際は0に2を加算していることになる)

図3 実体配線図

5)　表示が2になる
−6)　加減算切り替えを減算に切り替える
−7)　入力スイッチを3（=0011）に設定する
−8)　クロックボタンを押し3を減算する
−9)　表示が演算結果の−1になる

加算と減算の違いは加算（減算）する数をクロックによって入力する前にあらかじめ加減算切り替えを加算（または減算）に切り替えておくというだけのことです。上の手順を示した番号で＋または−の記号が頭に付いている6)～9)の部分だけ加算と減算で違う結果となっていますが，本質として手順に違いはありません。2つ以上の数の演算に関しても，

2−5+7=4

のような場合には，

1)　クリアボタンを押して表示を0にする
2)　加減算切り替えを加算にする
3)　入力スイッチを2（=0010）に設定する
4)　クロックボタンを押して2を入力する（実際は0に2を加算していることになる）
5)　表示が2になる
−6)　加減算切り替えを減算に切り替える
−7)　入力スイッチを5（=0101）に設定する
−8)　クロックボタンを押し5を減算する
−9)　表示が演算結果の−3になる
+6)　加減算切り替えを加算に切り替える
+7)　入力スイッチを7（=0111）に設定する
+8)　クロックボタンを押し7を加算する
+9)　表示が演算結果の4になる

以上のように最初に数を入力する手順は変わらず，あとは加算か減算かをその都度加減算切り替えスイッチで選択してから，演算する数を入力していく，ということを繰り返していけばよいことになります。ちょ

図4　スライドスイッチ

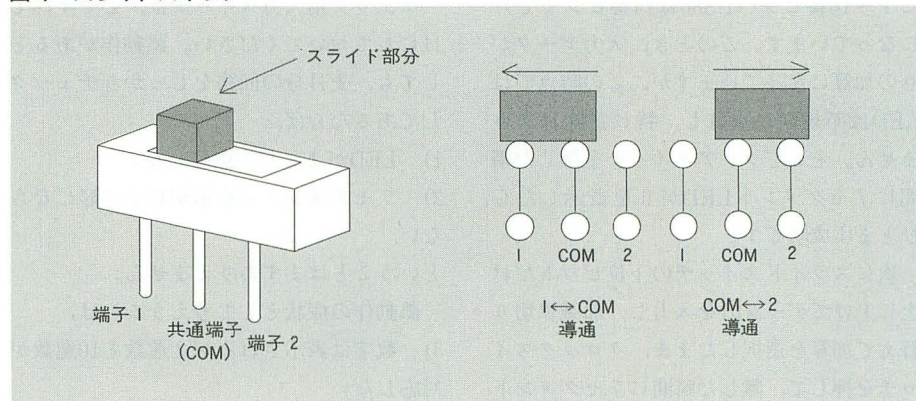
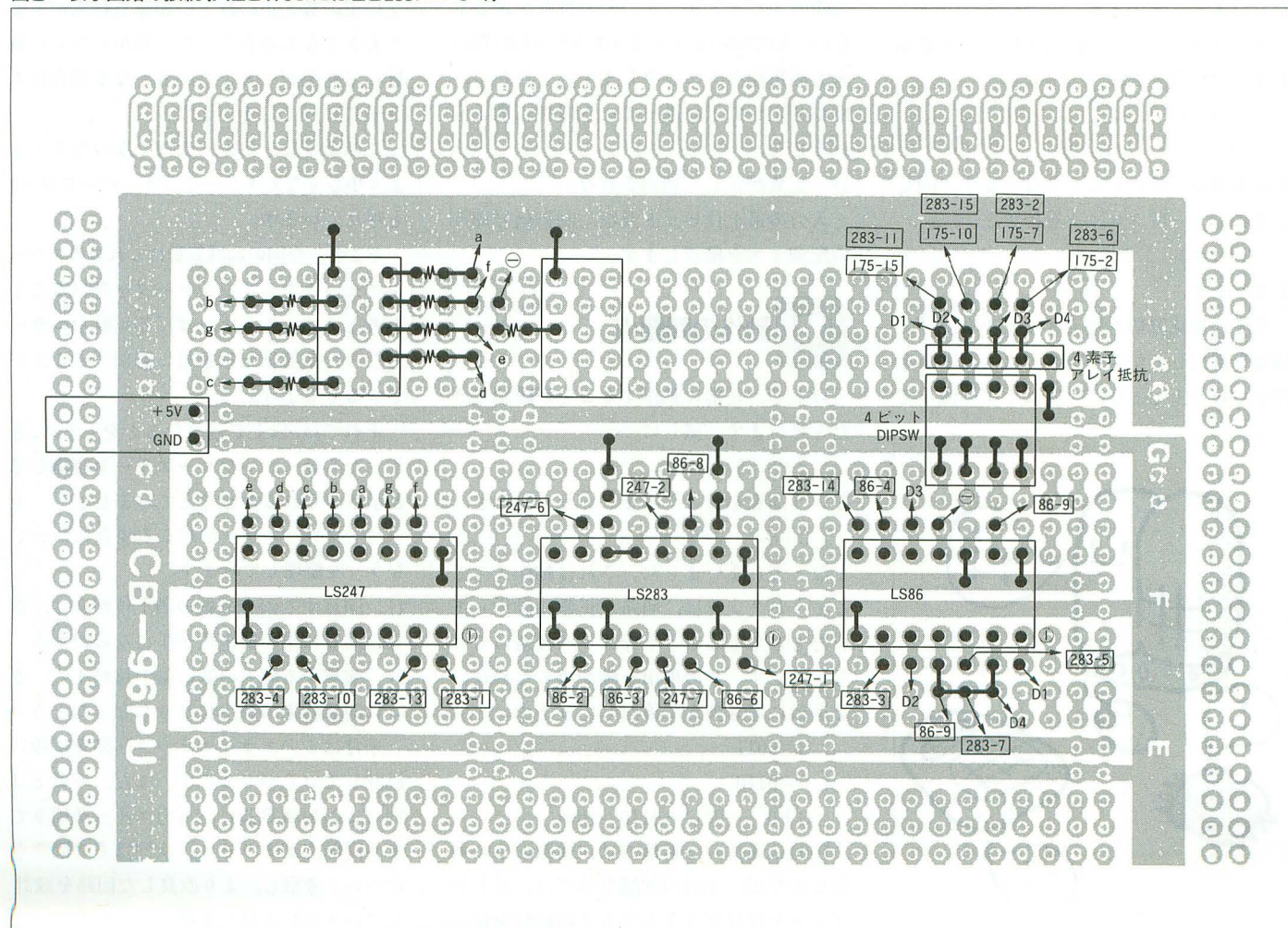



図5　表示回路の接続（ALS175またはLS283につなぐ）

うど手順の番号が同じように対応しているので，比較して見るとよくわかるでしょう。

動作チェック

LS86，LS283，ALS175をソケットに差し込み，まずスライドスイッチを全部下にして，電源をつないでみてください。なおTTL ICの向きは，ICを上から見ると切り欠きがあり，その切り欠きを基準に左回りに1～16番ピン（LS86は14番ピンまで）になっています。このとき，入力データが0の加算になっていますが，この時点ではLEDは不規則に点灯し，特に意味はありません。そこでクリアスイッチを押した瞬間に7セグメントLEDが0を表示したら，ひとまず成功です。

次にスライドスイッチの下位ビットだけ上に上げてデータ01を入力し，加減算切り替えで加算を選択したまま，クロックスイッチを押して，離した瞬間に7セグメントLEDが1を表示したらまず問題なく動作しています。そのままでもう一度クロックスイッチを押すと今度はLEDが2を表示します。このとき，

0+1=1

1+1=2

を順次演算していることになります。同じようにもう一度クロックを押すと，

2+1=3

となります。

今度は加減算切り替えスイッチを上げて減算を選択してから，クロックスイッチを押していきましょう。今度は1ずつ減っていきます。表示が0になってもさらにクロックを入力すると，左側の7セグメントLEDにマイナス記号「−」が表示され，全体で−1を表示していることになります。ここまでできれば，あとは，任意の加算が実現できますので，試してみてください。

illustration：Y.Kawahara

では，うまく動作しなかったときはどう対処するのがよいでしょうか。前回の表示回路はすでに正常な動作が確認されているものとして，次の段階に進んでいきます。

チェック用入力スイッチは，必ずすべてHにしておいてください。誤動作があるとしても，先月分の回路をしっかりチェックしてあるならば，

1)　LEDがまったく点灯しない

2)　7セグメントの表示が数字の形にならない

ということはまずありえません。

誤動作の症状として考えうるのは，

3)　数字は表示されるが2進数と10進数が対応しない

LS86とLS283，あるいはLS283とALS175の間の4ビットデータの桁の対応が正しくない。LS283のΣ1～Σ4出力やALS175の入力の桁順をチェックします。とにかく，桁がどこかで入れ違っている以外に考えられません。

4)　正負が正しく判別されない

入力の最上位ビット及び，制御信号周りの配線ミスを確認します。

動作の問題点

以上で，正常に動作することが確認できたと思います。試しに，

3+6=9

という演算を行ってみましょう。なんと，

3+6=−7

となってしまいました。これは動作ミスでしょうか？　実は，回路が正常に動作していたとしてもこのようなエラーは避けられないのです。この理由を確かめるために，2進数に置き換えて検討してみます。

3=0011

6=0110

3+6 → 0011+0110=1001

1001は通常の2進数では10進数の9に対応しますが，今回の加減算器では，最上位ビットを符号ビットとする2の補数を使っています。そのため，最上位が1である1001は負の数として判断されてしまうのです。ところで，2の補数表現では，

1001=−7

となっているので，

3+6=−7

という一見奇妙な結果になってしまうのです。

同様に，

−3−6=−9

という演算を試してみましょう。今度は，

−3−6=7

とまったく予想もつかない結果になってしまいます。これは，

−3=1101

−6=1010

−3−6 → 1101+1010=(1)0111

以上のように，最上位の繰り上がりは無視されてしまうので，演算結果は0111=7となるわけです。

今回の加減算器で扱っている2の補数の範囲は−8～7となっているため，このように正の数と正の数との加算で結果が7より大きくなる場合や，負の数からさらに減算して結果が−8より小さくなる場合はエラーになってしまいます。

このような，7より大きくなったり−8より小さくなるエラーを「オーバーフロー」と呼んでいます。

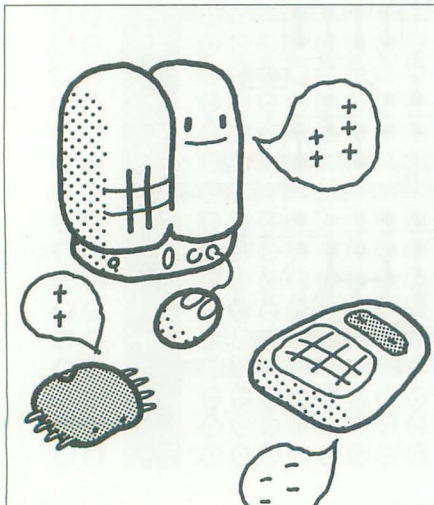
そこで，今回の加減算器にこれらのエラーをチェックするインジケータを付けることを考えます。そうすれば，不可解なエラーとなった演算結果を容易に識別することができるでしょう。

それでは，以上で述べたことをもとに，どうしたらオーバーフローエラーを認識して表示させられるかを考えてみましょう。オーバーフローエラーが起こる場合についてもう一度整理してみます。

1)　演算する両方の数の最上位が0で，さらに演算結果の最上位が1になったとき

2)　演算する両方の数の最上位が1で，さらに演算結果の最上位が0になったとき

来月はこのエラーチェック回路を実際に設計していくつもりです。また，それとは別にこの加減算器には入力スイッチ周りで多少問題点があります。次回はその問題点について考察し，より改良した回路を設計していきたいと思います。

Oh! X LIVE in '93

X68000・Z-MUSIC+PCM8用 (SC-55対応) FIGHTMAN

Inui Keisuke 乾 恵典

X1・MusicBASIC用 魔法のプリンセス ミンキーモモより 愛しのマーシカ

Yaguchi Yoshitaka 谷口 佳孝

春がやってきました。冬眠してた人も，さあさあ起きて！「春眠暁を……」なんて寝ぼけてないで，名曲ぞろいのLIVE inがアナタの入力を待ってますよ。と，いうことで今月も厳選の2曲のご紹介です。

Y・M・C・A?

それはYOUNGMENや。ってことで，今月の1曲目はX68000用に「FIGHTMAN」をお届けしましょう。何かのゲームミュージックと思ったキミ，ちょっと違うぞ。ジャパニーズフュージョンの大御所，カシオペアのアルバム「FULL COLORS」からの選曲なのです。演奏にはZ-MUSICシステムとPCM8.XとSC-55が必要なので気をつけてくださいね。

ちょっと豪勢なシステムかな？　とも思えますが，実際のところ内蔵音源はほとんど鳴っていません。サウンドエフェクトとして途中の歓声が鳴っているだけなので，PCM8.Xを持っていない人はSC-55だけでも聴けないわけではありません。

作品のデキは素晴らしいのひと言に尽きます。聴きはじめてすぐに「こりゃええわ」ってことで一発採用になってしまったほどです。さらに聴き込むほどに細かなテクが随所にちりばめられており，ゴージャスな仕上がりを見せてくれます。聴きどころとしては，歓声が入るあたりのギターとベースの掛け合い(ソロバトルともいう)や，田植え奏法が目に浮かぶようなオープニングのシンセフレーズですね。歯切れのよいサウンドも見事に再現されています。

SC-55クラスとしてはこれだけの音が出ていれば文句のつけようがありません。「ベースがもっと太い音だったらなぁ」とか，「CM-64みたいにギターカードが使えたらなぁ」なんて欲が出るほどいい感じです。

ちょっと褒めすぎかなって気もしますが，それはみなさんが実際に入力して聴いてみてから結論を出しても遅くはないでしょう。SC-55を持っている人はぜひとも入力すること。絶対にソンはしないよ。

FULL COLORS

ピピルマ・ピピルマ・プリリンパ

うむむ，ミンキーモモですか。昔のやつは見たことあるんだけどな。最近やってたやつは見てないんだよな。ってことで，魔法のプリンセス・ミンキーモモのオリジナルサウンドトラック「DaBaDaBa・DaBaDa」より「愛しのマーシカ」をお届けましょう。X1のMusicBASIC用です。

この曲は第7話の「雪の中のコンサート」で使われていたものです。話を要約すると，かけだしのピアニスト(スコタビッチ)が恋人(マーシカ)のために作った曲ですが，スコタビッチが貧乏やってるあいだにマーシカはほかの人のもとに嫁いでしまいます。長い年月が過ぎ，スコタビッチが巨匠と呼ばれるようになって「天国からこの曲が聴きたい」というメッセージがスコタビッチのもとに届く……というものです。スコタビッチは実は本当の作曲者である長谷川智樹さんだった……なんてオチはありません。

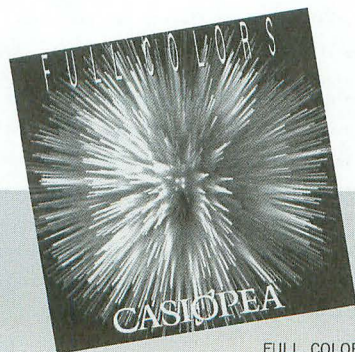
DaBaDaBa・DaBaDa

さて，作品に話を移しましょう。ストーリーに沿っているのでしょう。この曲はピアノで奏でられています。音色も4音使い，スタッカートやレガートごとに音色を切り替えるといった感じで使い分けていると思われます。

もちろん，デキのよさは保証済みなのですが，よりグレードアップをはかるためにもMIDIで演奏させてみたいですね。X68000ユーザーも移植に挑戦してみてはどうでしょう。リストも比較的短いほうだと思いますので，不可能ではないでしょう。注意点としてはMMLは2小節ごとに区切られていること，オクターブを逆向きにすることぐらいでしょうか。詳しくは，1992年10月号の囲み記事を参考にしてください。

それでは，パパレポ・パパレポ・ドリミンパ！

担当者より

「最近，LIVE inのページはセーラーうんちゃらだとか，なんとかムーンのネタが多くて軟弱だ！」というはげまし(?)のお便りをいただきましたが，断固否定させていただきます。「最近」ではなく，軟弱なの

は昔っからのことなんです。YOゼミの校歌やらQTハニー，ラジオ体操，伝説の昭和版ちびまる子ちゃんなどなど……思い起こせばLIVE inのページを飾った数々のイロモノ路線。ああ，軟弱。なかには予告が出たのに掲載されなかった作品もあります。ジャンルは問わずに誰の挑戦でも，という姿勢のため，セーラー服美少女戦士の挑戦も受けるのです。もちろん，氷河の戦士や秘密戦隊だって同様です。演歌やジャズでもハウスでも結構です。あなたの思い入れたっぷりの挑戦をお待ちしております。

それから3月号に掲載の「ケンのテーマ」ですが，印刷状態が悪くて打ち込みにくいものがあったようなので，ZMSファイルをもう一度掲載します。打ち込みに挫折してしまった人，ごめんなさい。ぜひぜひ再挑戦してくださいね。　(SIVA)

リスト1　FIGHTMAN


```
  1: .comment FIGHTMAN
  2: . ~ CASIOPEA [ FULL COLORS ] ~ by CHOCO 1993
  3:
  4: /----------------------------------------------------------
  5: /初期設定
  6:
  7: .adpcm_block_data=FIGHTMAN.ZPD
  8: /.adpcm_list=FIGHTMAN.CNF
  9:
 10:
 11: (i)
 12: (b0)
 13: (o132)
 14:
 15: /----------------------------------------------------------
 16: /トラック初期化
 17:
 18: (m1,1000)(aAdpcm,1)
 19: (m2,1000)(aAdpcm,2)
 20: (m10,4000)(aMidi1,10)
 21: (m11,4000)(aMidi2,11)
 22: (m12,4000)(aMidi3,12)
 23: (m13,4000)(aMidi4,13)
 24: (m14,4000)(aMidi5,14)
 25: (m15,4000)(aMidi6,15)
 26: (m19,4000)(aMidi10,19)
 27: (m20,4000)(aMidi11,20)
 28: (m21,4000)(aMidi12,21)
 29: (m22,4000)(aMidi13,22)
 30:
 31: /----------------------------------------------------------
 32: /MML定義
 33:
 34: / SC-55 system initalize
 35: (t10) @i$41,$10,$42
 36: (t10) x$10,0,$7f,0
 37:
 38: / Reverb : Room 1 ( Depth:127 , Time:80)
 39: (t10) x$40,$01,$30,0,0,0,127,80
 40:
 41: / Chorus : Feedback Chorus ( Depth:48 )
 42: (t10) x$40,$01,$38,4,4,48,0,64,3,30
 43:
 44: / Partial reserve
 45: (t10) x$40,$01,$10, 2, 3,4,4,4, 0,1,0,0, 0,2,0, 4,0,0,0
 46:
 47:
 48: /Guitar
 49:
 50: (t10) r4
 51: (t10) @30 i0 q8 @k0 p3 116 o4 @v127
 52: (t10) x$40,$11,$21,64,20,64,111,60,76,64,64,64,80
 53: (t10) x$40,$21,$10,88 / Bend pitch : 24
 54: (t10) r4
 55:
 56: (t10) u122 g8r8 q4|: a&(a,f),0r2..|q8r1 :| r2.g8r8
 57: (t10) |: <cr8crc>br a8.<d>r4 | br8brbar gr8e8.g8 :|
 58: (t10) br8brb<c#r d8r8q4e&(e,c)q8r8
 59:
 60: (t10) @37 o3 @v64 u100 x$40,$11,$32,60,92,64,14,70
 61: (t10) |: |: >q1'g8<c'<q6'g8<c'>q1'g8<c''g<c''g<c'< q6'a<d'r8
'a<d'r8>q1'a<d''a<d'
 62: (t10)        'a8<d'<q6'a8<d'r8 >q1'a<d''a<d'<<q6'dg'>q1'dg' '
dg'<q6'dg'r8>q1'dg''dg'
 63: (t10)    |  @v110>gg<q6gggar8 <u120'ea<d'r8'ea<d'>r4u100
 64: (t10)        >q1aa<q6aaabr8 <u120'dg<c'r8'dg<c'>r4u100@v64 :|
 65: (t10) | q6'f8.<cea''f<cea'&'f2.<cea'
 66: (t10)   'a8.<df''a<df'&'a2.<df' :|
 67: (t10) 'b-8.<da''b-<dg'&'b-2.<dg'
 68: (t10) @30 o3 @v127 u127 q8 x$40,$11,$32,60,76,64,64,64
 69: (t10) r4g8g8 <c8.e8.g8
 70:
 71: (t10) l8 <q4c&(c>c)r4 q8a.a.b <q4c&(c>c)r4 q8a-.a-.b-
 72: (t10) <q4c&(c>c)r4 q8g.g.a b2 a.b.<c
 73: (t10) q4e&(e>e)<r4 q8c#.c#.d q4e&(e>e)<r4 q8c.c.d
 74: (t10) e.d16&d2 r4 r1
 75:
 76: (t10) @37 o3 @v64 u100 116 x$40,$11,$32,60,92,64,14,70
 77: (t10) |: >q1'g8<c'<q6'g8<c'>q1'g8<c''g<c''g<c'< q6'a<d'r8'a<
d'r8>q1'a<d''a<d'
 78: (t10)     'a8<d'<q6'a8<d'r8 >q1'a<d''a<d'<<q6'dg'>q1'dg' 'dg'
<q6'dg'r8>q1'dg''dg'
 79: (t10) |  @v110>gg<q6gggar8 <u120'ea<d'r8'ea<d'>r4u100
 80: (t10)     >q1aa<q6aaabr8 <u120'dg<c'r8'dg<c'>r4u100@v64 :|
 81: (t10) >q6'f8.<cea''f<cea'&'f2<cea' r8'g2..<dg<c'r4
 82:
 83: (t10) @30 o4 @v127 u122 q8 x$40,$11,$32,60,76,64,64,64
 84: (t10) |: |: <cr8crc>br a8.<d>r4 | br8brbar gr8e8.g8 :|
 85: (t10) |  br8brb<c#r d8. q3e&(e8,c)>q8r8 :|
 86: (t10)    br8brb<c#r d8r8q4e&(e8.,c)
 87:
 88: (t10) @29 o2 116 u127 q1 @p80 @v116 x$40,$11,$32,60,92,60,11
4,64
 89: (t10) a8<@37a>@29aa<@37g>@29aa
 90: (t10) <@37f#&(f#,g)>@v100@29eee<@v116d8&(d,e)>
 91: (t10) a<@37a>@v100@29ee<@v116@37f#8&(f#,g)<@v127c> >@v100@29
ee<@v116@37g&(g,a)r4
 92: (t10) r1r1
 93: (t10) <q7'd8g'>>q1@v100@29ee<<@v116q7@37'd8g'>>q1@v100@29ee
 94: (t10) e<<@v116@37e>>@v100@29ee<<@v116@37d>>@29ee<<@37d&(d,e)
 95: (t10) >>@v100@29ee<<@v116@37d>>@v100@29ee<@v116@37g&(g,a)
 96: (t10) >@v100@29e<@v116@37e&(e8,f#)>@v100@29e<@v116@37f#&(f#8
,g)
 97: (t10) r1r2r4
 98:
 99: (t10) @30 o4 @z127 @v110 u127 q8 p3 x$40,$11,$32,59,88,65,64
,92
100: (t10) <d8&(d8,e) a4.&(a8,f)r4a8<c8
101: (t10) @b-683,0,0d.&@b0d4.&d32 c>u120a-u127gfedcd
102: (t10) @b-683,0,0e.&@b0e4.&e32 {dc>aa-ge}4d8ce
103: (t10) d4>u88q1d12d12<u127q8g12 fu120edu127cd&u88@b683du127@b
0<c>e
104: (t10) b8<@b-1024,0,0d32&@b0d32c&c4 r4>q5a8<e8
105: (t10) d8c#>a<rc#8c> rq3g@b683,0,0f32&@b0f32edc#c>a
106: (t10) b8<dc&c4 r4>a&u88@b1024au127@b0<ea
107:
108: (t10) <@b-1024,0,0d&@b0du120c#24c24>b24u127a24u120a-24g24u12
7e24e-24d24
109: (t10) u127dcd&u88@b341du127@b0>agf#f
110: (t10) e8.g<@b-1024,0,0e&@b0ega <deaef#4
111: (t10) r8a&u88@b683au127@b0f#ab <@b-683,0,0c&@b0c4..&(c>a)
112: (t10) r@b-683,0,0a32&@b0a32a<c>gaeg e-u120du127c>a<dc>ag
113:
114: (t10) <d12{dfa}6<@b-683,0,0c&@b0c&(c>b)a gfedc8>b<d
115: (t10) c8r8<@b-683,0,0c&@b0c8&(c>b) a4r8@b-683,0,0e&@b0e&
116: (t10) d8r8u127@b-683,0,0a&@b0a8. g4r8e8
117: (t10) f12{edc}6>agfe dfedcedc
118: (t10) >b-8&(b-8,g)g4 x$40,$11,$32,60,76,64,64,64
119: (t10) @v127 u127 <c8.e8.g8
120:
121: (t10) l8 <q4c&(c>c)r4 q8a.a.b <q4c&(c>c)r4 q8a-.a-.b-
122: (t10) <q4c&(c>c)r4 q8g.g.a b2 a.b.<c
123: (t10) q4e&(e>e)<r4 q8c#.c#.d q4e&(e>e)<r4 q8c.c.d
124: (t10) e.d16&d2 >a<c e.d.c d4r4>
125:
126: (t10) l16 u124
127: (t10) |: |: <cr8crc>br a8.<d>r4 | br8brbar gr8e8.g8 :|
128: (t10) |  br8brb<c#r d8. q3e&(e8,c)>q8r8 :|
129: (t10)    br8brb<c#r d8r8q4e&(e8.,c)
130:
131: (t10) @29 o3 @v48 u120 x$40,$11,$32,60,92,64,54,70
132: (t10) |:4 q1'>g<ce'q6|:3'g<ce':|'g8<ce'q1'>g<ce''>g<ce'
133: (t10)     q6|:4'b-<ea':|'b-8<ea'q1'>a<df#'q6'a<df#'
134: (t10)     q1'>a<df#'q6'a8.<df#'|:'a<df#'r:| |:'a-4<dg':| :|
135:
136: (t10) @30 116 o4 @v127 u127 q8 x$40,$11,$32,60,76,64,64,64
137: (t10) |: |: <cr8crc>br a8.<d>r4 | br8brbar gr8e8.g8 :|
138: (t10) |  br8brb<c#r d8. q3e&(e8,c)>q8r8 :|
139: (t10)    br8brb<c#r d8r8q4e&(e8.,c)
140: (t10) q8r4d8r8q4e&(e8.,c)r4 q8g8q4a&(a,f)r2.
141:
142:
143: /Brass
144:
145: (t11) r4
146: (t11) @62 i0 q7 @k0 p3 116 o4 @v122 u127 @i$41,$10,$42
147: (t11) x$40,$12,$21,64,20
148: (t11) x$40,$22,$10,88 / Bend pitch : 24
149: (t11) r4
150:
151: (t11) 'g#8<dg'r8 |: <'c8ea'>r2..|r1 :| r2 q8@b-8192,0,0'a#4<
cdg'q7@b0'a#8<cdg'r8<
152: (t11) |: 'eg<c'r8'eg<c'r'eg<c''egb'r 'e8.f#a''f#a<d'r4
153: (t11) |  'df#b'r8'df#b'r'df#b''df#a'r 'ceg'r8'>g#8.<ce''ceg'
r :|
154: (t11)    'df#b'r8'df#b'r'df#b'<c#>r 'gb<d'r8.'g8<ce'r8
155:
156: (t11) r1r1r1r2..ga
157: (t11) r1r1r1r2r8.<c8.d8>
158: (t11) r1r1r1r2.dega
159: (t11) r1r1r1r1
160:
161: (t11) r1r1r1r1
162: (t11) r1r1 r2.{ab<cdef}4> 'a<ceg''a<ceg'r2'a<ceg''a<ceg'r4
163: (t11) r1r1r1r2.dega
164: (t11) r1r1r1r2>q8@b-8192,0,0'a#4<cdg'q7@b0'a#8<cdg'r8<
165:
166: (t11) |:3 |: |: 'eg<c'r8'eg<c'r'eg<c''egb'r 'e8.f#a''f#a<d'r
4
167: (t11)         |  'df#b'r8'df#b'r'df#b''df#a'r 'ceg'r8'>g#8.<c
e''ceg'r :|
168: (t11)      |  'df#b'r8'df#b'r'df#b'<c#>r 'g8.b<d''g<ce'r4 :|
169: (t11)         'df#b'r8'df#b'r'df#b'<c#>r 'gb<d'r8.'g8b<e'r8
170: (t11) |1 |:32 r1 :| :|
171: (t11) |2 x$40,$12,$00,0,1 x$40,$12,$19,127 x$40,$13,$19,120
q8 / Piano (right hand)
172: (t11)    u127 reree-8e-8 d8dd8d8 d8d8dc#8c#8 c8.c8>a-8. b8.a
&a2r4
173: (t11)    u125<rd8.e8f8 g8.f&f4 u124>reu122a<cu127e8u120c>a <
u127d8.cr4
174: (t11)    >rd8u125du127e8f8 g8.<e8.e-8
175: (t11)    d8.er4 a4&aeu126e-8 u127d8d8e8f8
176: (t11)    l48 gu123fedc>bagu120fedc>u116bagf u110edc>u100bagf
e <<<
177: (t11)    x$40,$12,$00,0,61 x$40,$12,$19,122 x$40,$13,$19,124
 u127 116 q7 :|
178: (t11) |3 r4'gb<d'r8.'g8b<e'r4.@v127<g8a8r2.
179:
```

```
 180: (t12) r4
 181: (t12) @49 i0 q7 @k-16 p3 l16 o4 @v124 u127 @i$41,$10,$42
 182: (t12) x$40,$13,$21,12,20,64,64,80,64,14,64,59
 183: (t12) x$40,$23,$10,88 / Bend pitch : 24
 184: (t12) x$40,$13,$02,1 / Data is same as MIDI part 2
 185: (t12) r4
 186:
 187:
 188: /Piano + Strings
 189:
 190: (t13) r4
 191: (t13) @2 i0 q8 @k0 p3 18 o4 @v124 u127 @i$41,$10,$42
 192: (t13) x$40,$14,$36,64
 193: (t13) r4
 194:
 195: (t13) 'g#8<dg'r8 |: <'c8ea'>r2..r1 :|
 196: (t13) r1r1r1r1
 197: (t13) r4<er>a.a.<e> 'f#2ab<d'r<cde> 'c.egb''c16ea'r2. r1
 198: (t13) r4<er>a.a.<e> 'f#2ab<d'r<cde 'c.ea''c16eg'&'c2eg'>a<c>
 199: (t13) 'a-.<cdg''a-16<cdf'&'a-4<cdf'r2
 200: (t13) r4<er>a.a.<e> 'f#2ab<d'r<cde> '>b.<cegb''>a16<cea'r2.
r1
 201: (t13) r4<er>a.a.<e> 'f#2ab<d'r<cde> 'a.b-<dfa''g16b-<dg'&'g2
.b-<dg'
 202: (t13) '>b-2<dg''>b-.<eg''>b-16<eg'&'>b-4<eg'
 203:
 204: (t13) u117 q7
 205: (t13) x$40,$14,$36,72 x$40,$1c,$36,72
 206: (t13) 'c4efa<c'r4>'a.<efa''a.<efa''b<egb'
 207: (t13) '<c4dga-<c'r4'a-.<dfa-''a-.<dfa-''b-<dgb-'
 208: (t13) '<c4deg<c'r4'g.<deg''g.<deg''a<ea'
 209: (t13) 'b2<ce-f#b''a.<cea''b.<ceb'<'ceg<c'
 210: (t13) 'e4f#ab<e'r4'c#.fa<c#''c#.fa<c#''dfa<d'
 211: (t13) 'e4a-<ce'r4'c.a-<c''c.a-<c''da-<d'
 212: (t13) 'e.a<ce''d16a<d'&'d2a<d'r4
 213: (t13) x$40,$14,$36,64 x$40,$1c,$36,59
 214: (t13) 'c16fa''c16fa'r2'c16fa''c16fa'r4
 215:
 216: (t13) u127 q8
 217: (t13) r4<er>a.a.<e> 'f#2ab<d'r<cde> '>b.<cegb''>a16<cea'r2.
r1
 218: (t13) r4<er>a.a.<e> 'f#2ab<d'r<cde  'c.ea''c16eg'&'c2eg' c'd
2..ga-<c'r4
 219:
 220: (t13) |:24 r1 :|
 221: (t13) u117
 222: (t13) |: >r4'egb<ce'r '>b-.<dea''>b-.<dea''eab-<e'
 223: (t13)     'd2f#a<d' r'ca-<c''da-<d''ea-<ce'
 224: (t13) |  '>b.<cegb''>a16<cea'r4 r2 r1 :|
 225: (t13) 'a.<dfa''g16b-<dg'&'g2.b-<dg'
 226: (t13) r4>g4< c.e.g q7
 227:
 228: (t13) x$40,$14,$36,72 x$40,$1c,$36,72
 229: (t13) 'c4efa<c'r4>'a.<efa''a.<efa''b<egb'
 230: (t13) '<c4dga-<c'r4'a-.<dfa-''a-.<dfa-''b-<dgb-'
 231: (t13) '<c4deg<c'r4'g.<deg''g.<deg''a<ea'
 232: (t13) 'b2<ce-f#b''a.<cea''b.<ceb'<'ceg<c'
 233: (t13) 'e4f#ab<e'r4'c#.fa<c#''c#.fa<c#''dfa<d'
 234: (t13) 'e4a-<ce'r4'c.a-<c''c.a-<c''da-<d'
 235: (t13) 'e.a<ce''d16a<d'&'d2a<d''>a<fa''cfa<c'
 236: (t13) 'e.a<ce''d.fa<d''cfa<c''d2fa-b<d'
 237:
 238: (t13) |:8 r1 :|
 239:
 240: (t13) q8 l16 u117 x$40,$14,$36,64 x$40,$1c,$36,59
 241: (t13) |:r'>gb<ce':|'>g8b<ce'r8 '>b-8<ea''>b-<ea'|:'>b-8<ea':
|
 242: (t13) |:'>b-8<c#df#':|'>b-<c#df#'|:'>b-8<c#df#':| '>a-8.<dg'
'>a-8<dg'd8.
 243: (t13) '>g8.b<ce''>gb<ce'&'>g4b<ce' r'>b-8<dea'r'>b-8.<dea''>
b-<cdf'
 244: (t13) r'>b-8.<cdf'|:'>b-8<cdf':| '>a-8.<cdg''>a-4<cdg''>gb<c
e'
 245: (t13) r2 'e8.f#b-''ef#b-'r8.>b-< r2 '>a-8.<d''>a-<dg'&'>a-4<
dg'
 246: (t13) '>g8.b<ce''>gb<ce'r4 '>b-2<de'
 247: (t13) |:4'>b-8<c#df#':| '>a-4<cdg'r4
 248: (t13) |:10 r1 :|
 249:
 250: (t22) r4
 251: (t22) @49 i0 q8 @k-8 p3 18 o4 @v116 u127 @i$41,$10,$42
 252: (t22) x$40,$1c,$21,12,20,64,64,80,64,14,64,59
 253: (t22) x$40,$1c,$02,3 / Data is same as MIDI part 4
 254: (t22) r4
 255:
 256:
 257: /Organ
 258:
 259: (t14) r4
 260: (t14) @18 i8 q8 @k0 p3 l16 o3 @v127 u55 @i$41,$10,$42
 261: (t14) x$40,$15,$21,127,80,64,64,80,48,65,114
 262: (t14) r4
 263:
 264: (t14) r4 r1r1r1r2.'a#8<cdg'r8
 265: (t14) |: r4'a<ceg'r8. r4'a#8.<dea''a#<dea'
 266: (t14)     r4'b<df#'r8. 'g#<dg'r8.'g#8<dg'r8 :|
 267:
 268: (t14) |: r8'g<ce'rr4 'a#<de'r8.'a#8.<de''a#<de'
 269: (t14)  | r4'a<df#'r8. 'g#<cdg#'r8.'g#8.<cdg#''g#<cdg#' :|
 270: (t14) u65'a2b<df#' u55'g#<dg#'r8.'g#8.<dg#''g#<dg#'
 271: (t14) r4'g<ce'r4.. 'b-8.<de''b-<de'
 272: (t14) r4'a<df#'r8. 'g<cdg'r8.'g8.<cdg''g<cdg'
 273: (t14) r2 'fa<ce'r4..
 274: (t14) r2 'a-<cdg'r4..
 275: (t14) |: r4'g<ce'r8. 'b-<de'r8.'b-8.<eb-''b-<eb-'
 276: (t14)     r4'a<df#'r4.. 'b-8.<dg''b-<dg'
 277: (t14)     r4'g<ce'r8. 'b-<deb-'r8.'b-8.<deb-''b-<deb-'
 278: (t14)     u65'a2b<df#' u55'g#<dg#'r8.'g#4<dg#'
 279: (t14)     r4'g<ce'r8. 'b-<de'r8.'b-8.<de''b-<de'
 280: (t14)     r4'a<df#'r8. 'g#<dg#'r8.'g#8.<dg#''g#<dg#' r1r1
 281: (t14) |  r1r1r1r1 r1r1r1r1 :|
 282:
 283: (t14) |:4 r4'g<ce'r8. 'b-<deb-'r8.'b-8.<deb-''b-<deb-'
 284: (t14) |   r4'a<df#'r8. 'a-<da-'r8.'a-8.<da-''a-<da-' :|
 285: (t14)     r4'a<df#'r8. 'a-<da-'r8.'a-8<da-'r8 u24
 286:
 287: (t14) |:7 r8'a<cdg'rr'a<cdg'r8 'a8.<cdg''a<cdg'r'<dea''cdg'r
:|
 288: (t14)     r8'a<cdg'rr'a<cdg'r8 'a<cdg'r8.'<d4g<c' u55
 289:
 290: (t14) |:7 r4'g<ce'r8. 'b-<deb-'r8.'b-8.<deb-''b-<deb-'
 291: (t14)     r4'a<df#'r8.'a-<da-' r8.'a-8.<da-' 'a-<da-' :|
 292: (t14) |:10 r1 :|
 293:
 294: (t14) |: |:4 r4'g<ce'r8. 'b-<deb-'r8.'b-8.<deb-''b-<deb-'
 295: (t14)    |   r4'a<df#'r8. 'a-<da-'r8.'a-8.<da-''a-<da-' :|
 296: (t14) |      r4'a<df#'r8. 'a-<da-'r8.'a-8<da-'r8
 297: (t14)        |:8 r1 :| :|
 298: (t14)        r4'a<df#'r8. 'a-<da-'r4.. r1
 299:
 300:
 301: /Bass
 302:
 303: (t15) r4
 304: (t15) @38 i0 q7 @k0 p3 l16 o1 @v127 @i$41,$10,$42
 305: (t15) x$40,$16,$21,12,36,64,64,72,114,64,68,72
 306: (t15) x$40,$26,$10,88 / Bend pitch : 24
 307: (t15) r4
 308:
 309: (t15) u110e8r8
 310: (t15) |:4 u120q4a8a8<c8c8> u110q7f#8q1f#q7f#raa#8
 311: (t15)     u120q4b8b8<d8d8>|u110q7e8 q1e q7e reg8 :|
 312: (t15) u110q7<e8 q1e <u120q7@37e>> u110@38e4
 313:
 314: (t15) |:6 |4 u110q7f8f8f8f8 f8f8f8f8 b-8b-8b-8b-8 b-8<u120@3
7b->u110@38b-8.b-8 :|
 315: (t15)     u120q4a8a8<c8c8> u110q7f#8q1f#q7f#raa#8
 316: (t15)     u120q4b8b8<d8d8> u110q7e8 q1e q7e reg8 :|
 317: (t15)     u120q4a8a8<c8c8> u110q7f#8q1f#q7f#raa#8
 318: (t15) u120q4b8b8<d8d8 u110q7e8q4eee <@37e> q7@38(e8,g)
 319: (t15) u120q4g8g8 g8g8 u110gq7g8 <q4@37g> @38ff <@37f> @38f
 320: (t15) u120q4e8e8 f8g8 u110q7c8q1ccq7c8e8
 321:
 322: (t15) u120q4f8f8f8f8 u110q7ff<@37f8>@38ff<@37f8>@38
 323: (t15) >u120q4b-8b-8b-8b-8 u110q7b-b-<@37b-b->@38b-b-<@37b-b-
@38
 324: (t15) u120q4e8e8e8e8 u110q7ee<@37ee>@38ee<@37ee>@38
 325: (t15) u120q4e-8e-8e-8e-8 u110q7c8q1cq7<@37c>@38cc<@37c8>@38
 326: (t15) >u120q4b8b8b8b8 u110q7bb<@37bb>@38bb<@37b&(b<b)>@38
 327: (t15) u120q4e8e8e8e8 u110q7ee<@37ee>@38ee<@37ee>@38
 328: (t15) u120q4f8.f8fc&d f4g8f8 ggr8r4 r8ggr4>
 329:
 330: (t15) |:3 u120q4a8a8<c8c8> u110q7f#8q1f#q7f#raa#8
 331: (t15)   | u120q4b8b8<d8d8> u110q7e8 q1e q7e reg8 :|
 332: (t15) u120q4b8b8<d8d8> u110q7e8q4eee <@37e> q7@38(e8,g)
 333: (t15) u120q4f8f8f8f8 f8f8f8 u100q7b-8&b-2 u80(<e4>e)u100e4
 334:
 335: (t15) |:4 u120q4a8a8<c8c8> u110q7f#8q1f#q7f#raa#8
 336: (t15)     u120q4b8b8<d8d8>|u110q7e8 q1e q7e reg8 :|
 337: (t15) u110q7<e8 q1e <u120q7@37e>> u110@38e4
 338:
 339: (t15) @p48 q8u127a1&a2&a4 x$40,$16,$33,80 r8<@37u127(a8<g)
 340: (t15) ru127d&@b683dr@b0c&@b683c>@b0g&@b683g@b0
 341: (t15) |:>@38q1u120{ggg}8@37q8u127'g8<g'<:|
 342: (t15) x$40,$16,$36,96 u120q1{'<cg''<cg''b<f#''gd'}2. x$40,$1
6,$36,72
 343: (t15) q8@b0,-1024,0'a<e'&@b-1024,1024,0'a8<e'&@b1024,0,0'a<e
'@b0
 344: (t15) @38u127>a1&a2&a4 q7u100(<a4>>e)
 345: (t15) <<@37u110q8'>e8<b<e'q1'b<e''b<e'q8'b8<e'q1'b<e' x$40,$
16,$36,90 q4'a<d'
 346: (t15) r'a<d''a<d'r'a<d''a<d'r x$40,$16,$36,72 q8'g<c'
 347: (t15) r8'>e8<g<c''f#8b'> x$40,$16,$33,114 @38u110r4q8e8e4 p3
 348: (t15) |:4 u120q4a8a8<c8c8> u110q7f#8q1f#q7f#raa#8
 349: (t15)     u120q4b8b8<d8d8> u110q7e8 q1e q7e reg8 :|
 350: (t15) |:3 u120q4a8a8<c8c8> u110q7f#8q1f#q7f#rb-<@37u120b-8>@
38
 351: (t15)     u120q4b8b8<d8d8> u110q7e8 q1e q7e re<@37u120g8>@38
 :|
 352: (t15) u120q4<g8g8g8g8 u110q7gg<@37g>g@38ff<@37f>f@38
 353: (t15) u120q4e8e8u110q7f8g8 c8q1ccq7c8@37e8@38
 354:
 355: (t15) u120q4f8f8f8f8 u110q7ff<@37f8>@38ff<@37f8>@38
 356: (t15) >u120q4b-8b-8b-8b-8 u110q7b-b-<@37b-b->@38b-b-<@37b-b-
@38
 357: (t15) u120q4e8e8e8e8 u110q7ee<@37ee>@38ee<@37ee>@38
 358: (t15) u120q4e-8e-8e-8e-8 u110q7c8q1cq7<@37c>@38cc<@37c8>@38
 359: (t15) >u120q4b8b8b8b8 u110q7bb<@37bb>@38bb<@37b&(b<b)>@38
 360: (t15) u120q4e8e8e8e8 u110q7ee<@37ee>@38ee<@37ee>@38
 361: (t15) u120q4f8.f8fc&d f4g8f8 g8.g8.c&u88@b683cu120@b0>g4a-4
 362:
 363: (t15) |:4 u120q4a8a8<c8c8> u110q7f#8q1f#q7f#raa#8
 364: (t15)     u120q4b8b8<d8d8>|u110q7e8 q1e q7e reg8 :|
 365: (t15) u110q7<e8 q1e <u120q7@37e>> u110@38e4
 366:
 367: (t15) |:8 u120q4a8a8<c8c8> u110q7f#8q1f#q7f#raa#8
 368: (t15) |   u120q4b8b8<d8d8> u110q7e8 q1e q7e reg8 :|
 369: (t15)     u120q4b8b8<d8q7d&(d>b)u110e8r8e8r8
 370: (t15) <u100(c4>a)u120e8r8e8r8<u100(c4>a)
 371: (t15) u120g8a8r2.
 372:
 373:
 374: /Drums ( Snare & Bass )
 375:
 376: (t19) r4
 377: (t19) @1 @k0 p3 18 o2 @v110 u127 @i$41,$10,$42
 378: (t19) x$40,$10,$15,1 / Set drum kit 1
 379: (t19) r4
 380:
 381: (t19) dr 'de'cdr |:11 ccdr :| c4d4
 382: (t19) |:15 ccdr :| c.d.d
 383: (t19) |:15 ccdr :|
 384: (t19) '<c>c' {u80d u104d u127d}8 d <c>
 385:
 386: (t19) crdr |:11 ccdr :|
 387: (t19) c8.c8.<c8.>aa16ff
 388: (t19) d16d16r8c16a32<c32>f16c8.d16d16r4
 389:
 390: (t19) |:13 ccdr :| ccdc r2. u104d16u127d8.
 391:
 392: (t19) |:15 ccdr :| c4d4
 393:
 394: (t19) |:7 u127c4d4 u80<cc16>af16f :|
 395: (t19) u127c4d4c4d4
```

```
396:
397: (t19) |:31 ccdr:| '<c>c' {u80d u104d u127d}8 d16d16<c>
398:
399: (t19) crdr |:11 ccdr :| c.c16 {rrrr<cc crc>aaa raffrr}2.
400: (t19) 18 c.c16rccdd4
401:
402: (t19) |:15 ccdr :| crdr
403: (t19) crdr |:30 ccdr :| crdr
404: (t19) 116 <cc>acd4 d8{a<c>f}8c4
405: (t19) |:d*2d*22:| r2.
406:
407:
408: /Drums ( HiHat )
409:
410: (t20) r4
411: (t20) @1 @k0 @p36 18 o2 @v110 @i$41,$10,$42
412: (t20) x$40,$1a,$15,2 / Set drum kit 2
413: (t20) r4
414:
415: (t20) r4 |:15 r u48a-r u90a- :| u48a-4 u88b-4
416: (t20) |:7 r u48a-r u90a- :| r u48a-r u127a-
417: (t20) |:7 r u48a-r u90a- :| r2
418: (t20) |:8 r u48a-r u90a- :| u90a-a-ra-
419: (t20) |:6 r u48a-r u90a- :| r2
420:
421: (t20) u90 p3<c#> q2a-r @p36q8a-
422: (t20) |:11 rq2a- rq8a- :|
423: (t20) 116u70 p3<c#> a-a- <c#>
424: (t20) a-a-r@p36a- a-ra-rra-ra- r1
425:
426: (t20) 18 p3<u100c#> u48a-r @p36 u90a-
427: (t20) |:7 r u48a-r u90a- :| u90a-a-ra-
428: (t20) |:4 r u48a-r u90a- :| u90ra-r p3<u100c#2..r4
429:
430: (t20) c#> @p36 u64a-r u90a-r u64a- p3<u120e> u90a-
431: (t20) @p36 |:3 r u64a-r u90a- :|
432: (t20) r u64a- p3<u120e> u90a-
433: (t20) @p36 r u64a-r u90a-r u64a-r u120a-4
434: (t20) u64a-r u90a-r u64a- p3<u120e> u90a-
435: (t20) @p36 |:3 r u64a-r u90a- :|
436: (t20) @p36 r u64a- p3<u120e> u90a-
437: (t20) @p36 r u64a-r u90a- p3<u120e4c#4
438: (t20) 116 e4>r @p36u64a-a-a- ra-rra-rra-
439: (t20) |:6 u90a-u48a-a-a-ra-a-a-ra-rra-rra- :|
440: (t20) u90a-u48a-a-a-ra-a-a- a-a-a-8u88b-4
441:
442: (t20) 18 u90a-a-ra-
443: (t20) |:6 r u48a-r u90a- :| r u48a-r u88b-
444: (t20) |:5 |:3 r u48a-r u90a- :| r u48a-r u88b- :|
445: (t20) |:3 r u48a-r u90a- :| r4..u120a-16
446:
447: (t20) u90 p3<c#> q2a-r @p36q8a-
448: (t20) |:11 rq2a- rq8a- :|
449: (t20) 116 p3<u100c#8.c#> @p36u120{a-a-a-rrr a-rrrrr rrrra-r}
2.
450: (t20) p3<u100c#>u90a-a-<u100c#>u90a-a-<u100c#>u90a- <u120c#4
>b-4 18
451:
452: (t20) <c#> @p36 u64a-r u90a-r u64a- p3<u120e> u90a-
453: (t20) @p36 |:3 r u64a-r u90a- :|
454: (t20) r u64a- p3<u120e> u90a-
455: (t20) @p36 r u64a-r u90a-r u64a-r u120a-4
456: (t20) u64a-r u90a-r u64a- p3<u120e> u90a-
457: (t20) @p36 |:3 r u64a-r u90a- :|
458: (t20) @p36 r u64a- p3<u120e> u90a-
459: (t20) @p36 r u64a-r u90a- u120a-4b-4
460: (t20) p3<u100c#> @p36u90q2a-rq8a-
461: (t20) |:6 rq2a- rq8a- :| rq2a- rq8b-
462: (t20) |:7 rq2a- rq8a- :| rq2a-p3<u120eu88c#>
463:
464: (t20) r @p36 u64a-r u90a-r u64a- p3<u120e> u90a-
465: (t20) @p36 |:3 r u64a-r u90a- :|
466: (t20) r u64a- p3<u120e> u90a-
467: (t20) @p36 r u64a-r u90a-r u64a-r u120a-4
468: (t20) u64a-r u90a-r u64a- p3<u120e> u90a-
469: (t20) @p36 |:3 r u64a-r u90a- :|
470: (t20) @p36 r u64a- p3<u120e> u90a-
471: (t20) @p36 r u64a-r u90a- p3<u100c#4c#4 r4c#4e4r4 r1
472:
473:
474: /Drums ( Cowbell )
475:
476: (t21) r4
477: (t21) @1 @k0 p3 116 o3 @v48 @i$41,$10,$42
478: (t21) x$40,$1b,$15,2 / Set drum kit 2
479: (t21) r4
480:
481: (t21) r4 |:30 a-8a-a- :| r2
482: (t21) |:30 a-8a-a- :| a-4 r4
483: (t21) |:29 a-8a-a- :| a-4 r2
484: (t21) r4 |:23 a-8a-a- :| r1 r1
485:
486: (t21) |:27 a-8a-a- :| a-4 r1
487:
488: (t21) r4 |:28 a-8a-a- :| a-4 r2
489:
490: (t21) r1r1r1r1 r1r1r1r1
491:
492: (t21) |:61 a-8a-a- :| a-4 r2
493:
494: (t21) r4 |:23 a-8a-a- :| r1 r1
495:
496: (t21) r4 |:28 a-8a-a- :| a-4 r2
497: (t21) r4 |:29 a-8a-a- :| a-4 r4
498: (t21) r4 |:28 a-8a-a- :| a-4 r2 r1 r1
499:
500:
501: /Effect ( Gong )
502:
503: (t1) r2 p3 11 o1 @r1
504: (t1) r4 |:48 r :|
505: (t1) v5c&c&c2 c&c&c&c&c2.. v6c
506:
507: (t2) r2 p3 11 o1 @r1
508: (t2) r4 |:48 r :|
509: (t2) r v6c
510:
511: /-----------------------------------------------------------------
512: /演奏
513:
514: (p) / Thank you.
```

リスト2　FIGHTMANの音色コンフィグファイル

```
/for FIGHTMAN by CHOCO 1993

.olc=apl.pcm
```

リスト3　FIGHTMANのカウンタ表示

```
 1:00002AD8 00000000   2:00002610 00000000  10:00005010 00000000  11:00005010 00000000
12:00000060 00000000  13:00005010 00000000  14:00004F50 00000000  15:00005010 00000000
19:00005010 00000000  20:00005010 00000000  21:00005010 00000000  22:00000060 00000000
```

リスト4　愛しのマーシカ

日本音楽著作権協会（出）許諾第9272910-201号

```
10 '             Love Marsica
20 '                    from  MINKY MOMO
30 '
40 '    Composed by Tomoki Hasegawa
50 '
60 INIT:DEFINT A-Z:CLEAR &HFF00:TEMPO 0
70 DEFSTR A,B,C,D,E,F,G
80 DIM V(4,10),A(21),B(21),C(21),D(21),E(21),F(21),G(21)
90 '
100 "VOICE"
110 "MML"
120 FOR I=0 TO 21:PLAY A(I);:NEXT:PLAY ":";
130 FOR I=0 TO 21:PLAY B(I);:NEXT:PLAY ":";
140 FOR I=0 TO 21:PLAY C(I);:NEXT:PLAY ":";
150 FOR I=0 TO 21:PLAY D(I);:NEXT:PLAY ":";
160 FOR I=0 TO 21:PLAY E(I);:NEXT:PLAY ":";
170 FOR I=0 TO 21:PLAY F(I);:NEXT:PLAY ":";
180 FOR I=0 TO 21:PLAY G(I);:NEXT
190 PLAY ""
200 END
210 '
220 LABEL "VOICE"
230 MEM$(&HB190,36)=HEXCHR$("EC 00 71 71 31 31 14 02 19 04 5F 55
 5F 5A 05 09 03 0A 00 08 00 08 00 87 00 A7 00 00 00 00 00 00 80
00 00 00")
240 MEM$(&HB1B4,36)=HEXCHR$("E4 00 71 71 31 31 12 08 14 0A 5F 54
 5F 59 05 08 03 0A 00 06 00 06 00 87 00 A7 00 00 00 00 00 00 80
00 00 00")
250 MEM$(&HB1D8,36)=HEXCHR$("E4 00 71 71 31 31 12 08 14 0A 5F 54
 5F 59 05 07 03 09 00 03 00 03 00 87 00 A7 00 00 00 00 00 00 80
00 00 00")
260 MEM$(&HB1FC,36)=HEXCHR$("E4 00 71 71 31 31 12 08 14 0A 5F 54
 5F 59 04 05 02 07 00 03 00 03 00 77 00 97 00 00 00 00 00 00 80
00 00 00")
270 RETURN
280 '
290 LABEL "MML"
300 '
310 A( 0)="I1 O5 L8 V14 Q8 P3 K5 =0 T72 R
320 B( 0)="I1 O5 L8 V13 Q8 P3 K5 =0     R
330 C( 0)="I1 O4 L8 V13 Q8 P3 K5 =0     R
340 D( 0)="I1 O3 L8 V14 Q8 P3 K5 =0     R
350 E( 0)="I1 O4 L8 V8  Q8 P3 K5 =0     R
360 F( 0)="I2 O4 L8 V13 Q8 P3 K5 =0     R
370 G( 0)="I2 O4 L8 V13 Q8 P3 K5 =0     R
380 '
390 A( 1)="R@10A@7&AGAB-AGFE- D4V13CD&D@6T65V12D&T52D&T62D&T60D&
400 B( 1)="R@5D@5&D@7&D@91D@101& R1R@6 T78
410 C( 1)="B-@17&B-1& R1R@6
420 D( 1)="R@17R1 R1R@6
430 E( 1)="R@17R1 R2.R@6C4
440 F( 1)="R@10G@7&G2E-2 RG4.V11F+2&F+@6< V13
450 G( 1)="R@17R4F2.< A2&A&A@6V10A4.& A8&
460 '
470 A( 2)="R@8V14E-DC<T74B-T72B-AB->C T68D4&T60D4T50D4&T43D4&
480 B( 2)="R@4C2<F+2 A2G4&G@4V11F+4&> T75
490 C( 2)="G2D2 G2D2&D@8
500 D( 2)="R@8R1 R1
510 E( 2)="R@8R1 R1
520 F( 2)="R@8C4>C4C2< G4>G4<G4>V11C4&<<
530 G( 2)="R@8V13G2A2 B-2V11A2&
540 '
550 A( 3)="R@10A@7&AGAB-AGFE- T65D4.CT83D4.T75D&
560 B( 3)="R@5V13D@12&D@92E-@4&E-2< A1&>
570 C( 3)="B-@17&B-2B-2 R1
580 D( 3)="R@17R1 R1
590 E( 3)="R@17R1 R1
600 F( 3)="R@10V13G@7&G4>G4<E-4>G4< F4>A4<B-4>>D4&<
```


▶別にいまの処理速度に不満を感じることもないから，32ビットマシンなんかいらないよ。などと強がっても空しいだけの今日この頃。　吉田　務(20)大阪府

```
610 G( 3)="R@17V13D2<B-2> C4>E-4<FB-4.&
620 '
630 A( 4)="R@8E-DC<B-B-4AG T65V13G2T60V12F+2&> T75
640 B( 4)="R@4C@4&C1 R1
650 C( 4)="G@8&G1 R1
660 D( 4)="R@8R1 R1
670 E( 4)="R@8R1 R1
680 F( 4)="R@8C4>C4<<A2 D4>V12T60G4>V10D2&<<
690 G( 4)="R@8G2>C+2<< V12A4>A2.&
700 '
710 A( 5)="R@12V14A@7&AGAB-AGFE- D4CD&T72D@6V13D@90& T75
720 B( 5)="R@7D@12&D@92E-@4&E-2< RG4.V12F+2&>
730 C( 5)="B-@19&B-@88B-@8&B-2& R1
740 D( 5)="R@19R1 R1
750 E( 5)="R@19R1 R1
760 F( 5)="R@12V13G@7&G4>G4<E-4>E-4< A2D4>D4&
770 G( 5)="R@19V13D4<G4B-4E-4& E-2A4D4&
780 '
790 A( 6)="R@8V14T60E-T75DC<B-B-AB->C T78D2T70D4&T65D&T83D& T75
800 B( 6)="R@4V13C2<F+2 G2F+2&F+@4&>
810 C( 6)="G2C2 D2D2&D@8
820 D( 6)="R@8R1 R1
830 E( 6)="R@8R1 R1
840 F( 6)="R@8C2<D4>D4< V14G4>B-4V13D4>C4&<<
850 G( 6)="R@8R2A4D4> V14D4D4V13A4V12D4&
860 '
870 A( 7)="R@10AT83GAB-AGFE- T80D4.T73CT75D4.D&
880 B( 7)="R@5D@5&D@92E-@4&E-2< A4.&A@14F@10&F2&
890 C( 7)="B-@10&B-@88B-@8&B-2 R1
900 D( 7)="R@10R1 R1
910 E( 7)="R@10R1 R1
920 F( 7)="R@10G4>G4<E-4>E-4< F4>F4<B-4>B-4&<
930 G( 7)="R@10V13D4<G4B-4B-4> C4C4F4<B-4&
940 '
950 A( 8)="T78R@8E-DC<T73B-V13B-&T68B-T63AG T65G2V12F+2 V14
960 B( 8)="R@4G@4&G2&GC+4.& C2V12D2&>
970 C( 8)="E-@8&E-1 R2<I2V12A2&>> I1
980 D( 8)="R@8R1 R1
990 E( 8)="R@8R1 R1
1000 F( 8)="R@8C4>C4<<A2> D4>D4R4D4<
1010 G( 8)="R@8G4G2. A4>GRR2
1020 '
1030 A( 9)="T57B->G<B-DA>F+<AD B->G<B-DB>G<BD
1040 B( 9)="R16V14T75DD<GG>DD<F+F+> DD<GG>DD<GG16>
1050 C( 9)="R4G4R4F+4 R4G4R4G4 V14
1060 D( 9)="R1 R1
1070 E( 9)="R1 R1
1080 F( 9)="I3 G4>B-4<D4>F+4< G4>B-4<G4>G4<
1090 G( 9)="I3 D4D4<A4A4> D4D4D4D4<
1100 '
1110 A(10)="G>>GFE-C+<GAB- A1 T57
1120 B(10)="R@4V13E-@20DC<B-B-4AG& G@4G@68GT70V12F+4.T67V10F+> V
14
1130 C(10)="R2<V12C+V15C+4.& CV14C&C2.
1140 D(10)="R1 R1
1150 E(10)="R1 R1
1160 F(10)="C4>C4<V12AV14A4.& AV15>E4>E-V14D<V13AF+D<
1170 G(10)="G4G&G2& G4>V15A4<V14D2& D>V13
1180 '
1190 A(11)="DT78B-D<G>DAD<F+> DB-D<G>DBT70D>T52!127D& T75
1200 B(11)="R16T75GG<B-B->F+F+<AA> GG<B-B->GG<BT63B16&>
1210 C(11)="V12 B->D4DCD4D< B->D4D<B>D4<G> V14
1220 D(11)="R1 R1
1230 E(11)="R1 R1
1240 F(11)="G4>G4<D4>F+4< G4>G4<G4>G4
1250 G(11)="D4D4<A4A4> D4D4D4D8<
1260 '
1270 A(12)="R@4DCC<B-B-AT70AG T72V14G&G@4V13C+T70EGT65F+T60V12CD
F+& F+@6V13
1280 B(12)="R@2V13G@2&G1 G@4<V11A4G4F+2& F+@3>V13D@3&
1290 C(12)="E-@4&E-1& E-@4R1<V13B-@6&
1300 D(12)="R@4>RGAB->!127C+<B-AG< R@4R1R@6 V13
1310 E(12)="R@4R1 R@4R1R@6 V13
1320 F(12)="I4V15E-@4&E-2.<A4& A@4>I3V12G2D4>D4& D@6<<
1330 G(12)="I4V15E-@4&E-1& E-@4R4>I3V12E4.A4D4& D@6 V13
1340 '
1350 A(13)="T65AT70GT75V14AB-V15AGFE- V14D4T68CT65DD2 T75
1360 B(13)="D2V15E-2 V14R@188C@4&
1370 C(13)="B-2V15R2 V14R@184G@8&
1380 D(13)="B->G>V14D<GV15RE-B-E-< V14G>CGCRCAD<
1390 E(13)="R16DB-V14B-D<V15B->GG<B- V14A>DDR<A>DF+C
1400 F(13)="G4>G4<V13E-4>E-4< A2D4>D4
1410 G(13)="D4<G4V13B-4E-4& E-2A4D4&
1420 '
1430 A(14)="E-DC<T70B-&B@4T67B-@20AB->C T75V15DB-D<T70G>T65DV14A
D<V13F+> V14T60
1440 B(14)="C2&C@2<F+@94 >R16V15GG<B-B->V14F+F+<AV13A@10>V14D@2&
1450 C(14)="G2D2 R@188B-@4&
1460 D(14)="G>E->C<E-V13RD>D<F+ V14G2F+2<
1470 E(14)="CGGCV13CF+B-C< R1V14
1480 F(14)="C2<D4>D4< G4>B-4D4>C4<<
1490 G(14)="D4>C4<A4D4> D4D4A4D4
1500 '
1510 A(15)="AT75GAB-&B-@4A@20GFE-& E-@4D@44CC&C@2D@46DD
1520 B(15)="D2E-2< A2F2
1530 C(15)="B-1 R1
1540 D(15)="G>D>D<G<RB->B-E- RCFF RCFF
1550 E(15)="B->GB-D<G>E-G<B- A>E-AC16&C@10<F@2&F>DA4<
1560 F(15)="G4>G4<E-4>E-4< F4>F4<B-2>
1570 G(15)="D4<G4B-4B-4> C2.<B-4
1580 '
1590 A(16)="R@4E-@20DC<B-T68B-4AG T65G2F+2> T63
1600 B(16)="G2R2 D2D@94>V12D@2&
1610 C(16)="R1 R@188V12B-@4&
1620 D(16)="RCRE-RC+RC+ RCADRCAD<
1630 E(16)="G>E-GC<G>EE<A A>DGC<A>DF+C<
1640 F(16)="C4C4<A2> D4D4D4D4<
1650 G(16)="G4>C2.< A4>G2.
1660 '
1670 A(17)="AT75GAT70B-AT75GFE- D4T72CD&T75D@4D@92&
1680 B(17)="D4V14D4E-2< R2F+2&
1690 C(17)="B-1 R1
1700 D(17)="G>DB-DRE-B-E-< RG>C<GRA>F+<A
1710 E(17)="B->GG<B-B->GG<B- E-AAE-F+>DD4<
1720 F(17)="G4>G4<E-4>E-4<< A2>D4>D4
1730 G(17)="D4<G4B-4E-4& E-2A4D4&
1740 '
1750 A(18)="D@4E-@20DC<B-B-AB->T70C T75DB-D<G>T70DAD<F+@10&F+@8>
1760 B(18)="F+@2>C@94<F+2& F+16>GG<B-B->F+F+<A16&A@10&A@4>D@4&
1770 C(18)="G2D@88G@8& G2F+4&F+&F+@10T75B-@8&
1780 D(18)="RG>E-<GRA>F+<A> R2.RR@10R@8
1790 E(18)="E->CC<E-F+B->D@32B-@4& B-2A4.&A@10R@8< R16
1800 F(18)="C2<D4>D4< G4>G4<D4>F+8&F+@10&F+@8<
1810 G(18)="R2A4D4> D4<G4>D4C8&C@10&C@8
1820 '
1830 A(19)="AGAB-AGFE- D4CDD4&T70DV13D&
1840 B(19)="D2E-2< A2B-2&
1850 C(19)="B-2B-2 R1
1860 D(19)="RDB-DRE-B-E- RC>C<F<B->F>T72F<T70B-
1870 E(19)="B->GG<B-B->GG<B- A>FACDB->D<F<
1880 F(19)="G4>G4<E-4>E-4< F4>F4<B-2>
1890 G(19)="D4D4<B-4B-4> C4C2<B-&
1900 '
1910 A(20)="D@4E-@20T73DC<B-B-4T70AG G2F+2
1920 B(20)="B-@2B-@190 D@190D@2&
1930 C(20)="G1 R@188<B-@4&
1940 D(20)="RCGC<RG>G<B-> RDAT68D<T62F+>C>V15T45D<T40GV14
1950 E(20)="G>E-E-<GEB->C+<G A>EE<AA>F+V11AT32V10DV14
1960 F(20)="R4C2. D4D4F+2<
1970 G(20)="B-2>C+2& C+<A4>G2.
1980 '
1990 A(21)="V14G1>> V15D1
2000 B(21)="D1> V15D1
2010 C(21)="B-1 R1
2020 D(21)="T40RT60G>D<B->GDT50B-G V15G1
2030 E(21)="T50DT69B-G>D<B->GT40DB-16 V15B-1
2040 F(21)="G4>G4B-4>G4<< V15G1
2050 G(21)="D4A4>D4A8 R1
2060 '
2070 RETURN
```

リスト5　ケンのテーマ（3月号に掲載のものと同じ）

```
 1: /--------------------------------------------------------
 2: /                 STREET FIGHTER 2 AMERICA(KEN)
 3:
 4: /                   composed by ALFH LYRA
 5:
 6: /                      1992 / 1 / 10
 7:
 8: /                    programmed by NAKA
 9: /--------------------------------------------------------
10: .comment STREET FIGHTER 2 AMERICA(KEN) programmed by NAKA
11:
12:                             / ～おことばタイム～
13: (i)                         / スト2の音楽の中ではではこれが一番
14: (b1)                        / イカス（死語）よね！それを今回はバ
15: (o182)                      / リバリ（死語）のロックにアレンジし
16:                             / てみました。アレンジしたというより、
17: (m1,2000)(amidi1,1)/ スト2のアレンジ曲"春雷"の中にちょっ
18: (m2,2000)(amidi2,2)/ とだけ健？(KEN) の曲が入っていたの
19: (m3,2000)(amidi3,3)/ ですが、それがバリバリ（死語）のロッ
20: (m4,2000)(amidi4,4)/ クにアレンジされていて猛烈（死語）
21: (m5,2000)(amidi5,5)/ に気に入ってしまったのです。
22: (m6,2000)(amidi6,6)/ ということで作ってみました。
23: (m7,2000)(amidi7,7)/ アレンジだから嫌だぁーなんて言わない
24: (m8,2000)(amidi10,8)/  て聴いてみそ！
25: (m9,2000)(amidi10,9)
26:
27: /---- 初期化 ------------------------(ショキカスルナンテショーキカ！)￣_￣;
28:
29: .roland_exclusive $10,$42={$40,$00,$7f,$00}
30:
31: /        REVERB&CHORUS SET
32: .roland_exclusive $10,$42={$40,$01,$30
33:                             $03,$03,$00,$4d,$12,$00,$00
34:                             $02}
35:
36: /        PART 4  TONE  SET
37: .roland_exclusive $10,$42={$40,$14,$30
38:                             $50,$45,$72,$58
39:                             $40,$72,$40,$40}
40: /        PART 5  TONE  SET
41: .roland exclusive $10,$42={$40,$15,$30
42:                             $50,$45,$72,$58
43:                             $40,$72,$40,$40}
44: /        RESERVE SET
45: .sc55_v_reserve $10={1,2,4,3,0,0,0,0,0,1,0,0,0,0,0,0}
46:
47: /---- BASS ------------------------------------------------
48: (t1) r4n1@34@u127v13_2q8@p56o218@k-3 @i$41,$10,$42 @e50,20
49:
50: f1&f2cfe-r e-4d-rd-1 rd-c>q7b-4|:14b-:|<c4|:14c:|[do]
51: |:f4|:7f :|rf f rf e-4 d-4|:7d-:|rd-d-rd-c4>b-4
52:      |:7b-:|rb-b-rb-a-b-<c4|:7c :|rc c rcc|c:|rr
53: 14|:|:d-.|d-.d-:|d-d-8d-|>|:b-.|b-.b-:|b-b-8b-8<18
54: f4|:7f:|rffe-fe-4r:|
55: 14|:e-.|e-.e-:|e-e-8e-8 18c4cfa-<c>b-a-<cr>crccer
56: |:f4 |:7f :|rffrfe-4d-4     |:7d-:|rd-d-rd-c4>|
57:   b-4|:7b-:|rb-b-rb-a-b-<c4|:7c :|rcccccr:|
58:   e-4|:6e-:|e4|:6e:|f&f1&f2..
59:
60: /---- MELODY ----------------------------------------------
61: (t2) r4n2@63@u120v15q8p3o518 @i$41,$10,$42@e110,30@h42@m80
62:
63: f1&f2cfe-r e-4d-rd-1&d-d-c>b-1&b-2..b-<c&c1&c1[do]14o5
64: |:|:3rcfb-8|a-g8f4.e-8r8f8:|<c4d-8e8>|
```

▶３月号のSTUDIO Xの岩瀬さん。入浴前にポリデントを入れるとどうなるか教えてください。私は寮なので，勝手に風呂へポリデントを入れられないのです。もしかして，体を洗わなくてもある程度すべすべするのですか？　　守田　孝雄(28)京都府

```
 65: b-a-8b-8<c2^8>b-a-gf8e2^8:| b-4a-8g8<c8r>b-2a-ga-8g2^8@m90
 66: v14^2@57o418|:f4.g^2a-4.b-4a-ge-d-4.<c^2&c2.>b-<c&c*384>:|
 67: ^3@m80@63o514rcfb-8<c4.>18b-a-rgra-4.gfre-rf4 r4fga-2
 68: rb-4a-4gb-rb-a-4g4a-14g.feg8<c8r8|:5c8:|r8>
 69: <c.>b-a-b-8a-.gf8ca-.gf8e-frf8g8a-2r8b-<c
 70: 18d-crc>b-4a-reg4.f1r2.v13
 71:
 72: /---- BACK -------------------------------------------
 73: (t3) r4n3@49@u116v12q8@p74o518@i$41,$10,$42@e55,10@h42@m60
 74:
 75: 'a-<cf'288'a-<c''a-<cf''gb-<e-'
 76: r'g4b-<e-''fa-<d-'r'f1a-<d-'
 77: r'fa-<d-''e-g<c''d-fb-'360'd-fb-''eg<c'384[do]@49o5v11^2
 78: |:'fa-<c'384'fa-<d-'384'fb-<d-'384|'g<ce'384:|'g<ce'240
 79: @62v14o5|:5'eg<ce':|r4@92o5v11
 80: |:|:c+fa-<c>|r2:|rb-a-g||:>b-<d-fa-|r2:|rra-gr1r1:|
 81: |:e-gb-<e->|r2:|r<d-c>b-r1r2..v10@49o5@u127
 82: |:'fa-<c'384'fa-<d-'384|'fb-<d-'384'g1<ce'
 83: r4@62v14o5|:5'eg<ce':|r@49v11o5:|
 84: ^2'g1b-<e-''g1b-<e''a-<cf'384
 85:
 86: /---- GUITAR ------------------------------------------
 87: (t4) r4n4@31@u124v10q8@p44o418@i$41,$10,$42
 88:  @e75,80@h42@m60
 89:
 90: 'cf'288>'a-<c'<'cf'>'b-<e-'r'b-4<e-''a-<d-'r'a-1<d-'
 91: r'a-<d-''g<c' @q1
 92: |:'f4b-'|:'fb-''b-<f''fb-':|:|
 93: |:'g4<c'|:'g<c''<gc''g<c' :|:|[do]o3v10^2
 94: |:|:'f4<c' |:'f<c''<cf''f<c' :|:|o3
 95:    |:'a-4d-'|:'a-d-''d-<a-''a-d-' :|:|
 96:    |:'b-4f' |:'b-f''b-<f''b-f' :|:||o3
 97:    |:'g4<c' |:'g<c''<gc''g<c' :|:|:|
 98:      'g4<c' |:'g<c''<gc''g<c' :|
 99:      rr^5|:5'g<c':|_9r4 o3
100: 'a-d-'384'f>b-'360|:|:'f4<c'|:'f<c''<fc''f<c' :|:|'fc'
101: 'a-d-'384'b-e-'360|:|:'g4c'|:'gc''g<c''gc' :|:|^5
102: |:|:'f4<c' |:'f<c''<cf''f<c'     :|:|o3
103:    |:'a-4d-'|:'a-d-''a-<d-''a-d-':|:||
104:    |:'b-4f' |:'b-f''b-<f''b-f'   :|:|
105:      'g4<c' |:'g<c''<gc''g<c'    :|
106:   q6'g4<c'q7^3|:5'g<c':|r_3         :|@q1
107:   'b-4<e-' |:'b-<e-''<b-e-''b-<e-':|
108:   'b-4<e ' |:'b-<e ''<b-e ''b-<e ':|
109:   o4'cf'192rr @e110,70|:'cf':|r|:'cf':|r@e75,80
110:
111: /---- GUITAR2 -----------------------------------------
112: (t5) r4n5@30@u120v9q8@p84o318@i$41,$10,$42 @k5
113:  @e75,80@h42@m60
114:
115: 'cf'288>'a-<c'<'cf'>'b-<e-'r'b-4<e-''a-<d-'r'a-1<d-'
116: r'a-<d-''g<c'
117: |:q8'f4b-'q6|:'fb-''b-<f''fb-':|:|
118: |:q8'g4<c'q6|:'g<c''<gc''g<c' :|:|[do]o2v9
119: |:|:q8'f4<c' q6|:'f<c''<cf''f<c' :|:|o2
120:   |:q8'a-4d-'q6|:'a-d-''d-<a-''a-d-' :|:|
121:   |:q8'b-4f' q6|:'b-f''b-<f''b-f' :|:||o2
122:   |:q8'g4<c' q6|:'g<c''<gc''g<c' :|:|:|
123:     q8'g4<c' q6|:'g<c''<gc''g<c' :|
124:     rr^5|:5'g<c':|_9r4 o2q8
125: 'a-d-'384'f>b-'360|:|:q8'f4<c'q6|:'f<c''<fc''f<c':|:|'fc'
126: q8'a-d-'384'b-e-'360|:|:q8'g4c'q6|:'gc''g<c''gc' :|:|^5
127: |:|:q8'f4<c' q6|:'f<c''<cf''f<c'     :|:|o2
128:   |:q8'a-4d-'q6|:'a-d-''a-<d-''a-d-':|:||
129:   |:q8'b-4f' q6|:'b-f''b-<f''b-f'   :|:|
130:     q8'g4<c' q6|:'g<c''<gc''g<c'    :|
131:     q4'g4<c' q6^3|:5'g<c':|r_3       :|
132:   q8'b-4<e-'q6 |:'b-<e-''<b-e-''b-<e-':|
133:   q8'b-4<e 'q6 |:'b-<e ''<b-e ''b-<e ':|
134:   q8o3'cf'192rq6r@e110,70|:'cf':|r|:'cf':|r@e75,80
135:
136: /---- DRUMS -------------------------------------------
137: (t8) r4n10@17@u115v13 o214 @i$41,$10,$42 @e30,20
138:
139: |: crrr8 c8c8rr8c8d8|c:|c8c818
140: rcdrccdc rcdrccdr ccdrccdc rcdrccdc[do]
141: |:8rcdrcc|dc:|dc
142: |:7rcdrcc dc:|c8@r1|:5'd<d>':|r4@r014
143: |:|:c.c.dc.|c.@u127<d8>b8@u115:|c@u127<d8>b8@u115c8
144: 18rcdrccdc rcdrccd|d:|c18
145: rcdrccdc rcdrccdc |:3rcdrccdc|rcdrccdc:|cddd<d>bcc
146: rcdrccdc rcdrccdc |: rcdrccdc rcdrccdc:|14rccc cc18ddcc
147:
148: /---- HAT ---------------------------------------------
149: (t9) r4n10 @u127 o214
150:
151: |:'<c+a'rrr rrr8<c+8|a>:|18
152: @u127<a>@u110|:16f+:|@u127<c+>@u110|:15f+:|[do]
153: |:4@u127<c+>@u110 |:7f+:|a+|:7f+:|
154:          <a >      ||:7f+:|a+|:7f+:|:||:7f+:|r|:5a+:|r4
155: |:r<c+>f+f+a+4.r|:3 rf+f+f+a+4.r:||:6f+:|a+a+4|:7f+:|:|
156: |: @u127<c+>@u110 |:7f+:|a+|:7f+:|
157:     @u127<a >@u110 |:7f+:|a+|f+|:6f+:|:|f+|:5a+:|f+
158:     @u127<c+>@u110 |:7f+:|a+|:7f+:|
159:     @u127<a >@u110 |:7f+:|a+|:7f+:|
160:     @u127<c+>@u110 |:7f+:|@u127<a>@u110|:7f+:|
161:     @u127<'c+a'>r1r2..
162:
163: /---- LOOP --------------------------------------------
164:
165: (t1) [loop]/ SC－55はいいですよ！何が良いのかだって？
166: (t2) [loop]/ 音が良いんです。あの価格でこの音質が楽しめる
167: (t3) [loop]/ なんて。MO－32なんかよりはずっといいです。
168: (t4) [loop]/ 今回のプログラムでは本当はFMも鳴らそうとし
169: (t5) [loop]/ てたんたけど、そんな必要はありませんでした。
170: (t8) [loop]/ だって、音が良いんだもん！
171: (t9) [loop]/                        BY もとMTユーザー
172:
173: (p)
```

(善)のゲームミュージックでバビンチョ

●THE NINJA WARRIORS COMPLETE ALBUM/TAITO ZUNTATA　CD:PCCB-00116

ポニーキャニオン　2,500円(税込)　3/19発売

6年の歳月を越えて，いまだに人気の絶えないゲームミュージック「忍者ウォーリアーズ」のフルアレンジアルバムが発売される。全曲ともオリジナル曲のイメージを残したスマートなアレンジで聴きやすい。人気曲の「DADDY MULK」や「ARE YOU LADY?」もシンセサイザ主体のアレンジで変に生演奏を意識したアレンジでなくてひと安心(?)。さすがキーボーディスト中心のZUNTATAといったところか？　曲の合間にYMOスネークマンショー風にショートコントが入る。私もスト・パーは結構ハマってるほうです(CD参照)。ギャハハ。

お勧め度　9

●ナムコゲームサウンド・エクスプレス VOL.8　CD:VICL-15018

ビクター音楽産業　1,500円(税込)　3/24発売

コズモギャングズのビデオゲーム「コズモギャング・ザ・ビデオ」と「コズモギャング・ザ・パズル」のオリジナルサウンドアルバム。賑やかなBGMとS.E.は，ゲームを知らなくてもウキウキさせてくれる。特にボーナスステージのダンステーマはパロディ曲もちらほら，思わずニヤッとさせられるものもあり。

お勧め度　7

●KONAMI GM HITS FACTORY I　CD:KICA-1122

キングレコード　2,800円(税込)　3/24発売

6，7年近く前にアルファレコード/GMOレーベルから発売された4タイトルより，アレンジバージョンを9曲，さらにゲームソフトのCMなどで使用された曲を2曲収録した，全曲アレンジバージョンのアルバム。ゲームミュージックのアレンジバージョンのなかでは当初からお気に入りだった「グラディウス」メドレー，「ツインビー」メドレーも収録されていてご機嫌。うーん，聴いてみて思ったんだけど，ゲームミュージックのアレンジコンセプトってこの頃のほうが洗練されていたと思うんだけど，気のせい？

お勧め度　9

●エルムナイト　CD:DPCX-1009

ポリスター　2,400円(税込)　3/25発売

「ハイビジュアルSF3Dバトル」ゲームと仰々しいサブタイトルで世に送り出されたロボットゲームのオリジナルBGMアルバム。FM TOWNS/PC-9801版の2バージョンを40曲収録。FM TOWNS版はCD-ROMからの収録のため音質とアレンジはハイレベルでボーカル曲もあり。PC-9801版はそろそろ音色セットに飽きがきた感があるものの，音源スペックを超えたサウンドを聴かせてくれる。ゲームのほうはX68000への移植が楽しみといったところ。

お勧め度　7

●熱唱!! STREET FIGHTER II -CAPCOM-
インスト編　CD:PCCB-00114
オリジナルカラオケ編　CD:PCCB-00115

ポニーキャニオン　各2,000円(税込)　3/19発売

あのブッ飛びボーカル版ストIIミュージックが，別アレンジで再び我々の前に姿を現わした。インスト版は，メロディラインがギター，サックスなどの楽器に置き換わっている。カラオケ版のほうは歌詞カード入りで，突然のお客様とカラオケパーティになってしまっても安心。バッフーーン。

お勧め度　6

●DRAGON GUN/DATA EAST GAMADELIC　CD:PCCB-00117

ポニーキャニオン　1,500円(税込)　3/19発売

「ドラゴンボール」の悟空の声役の野沢雅子がおしゃべりするファンタジータッチのガンシューティングゲーム「ドラゴンガン」のオリジナルBGMアルバム。PCM音源を駆使した映画音楽のようなシンフォニックなサウンドがいっぱい詰め込まれている。S.E.集は国内/国外版の両方を収録。

お勧め度　7

＊　＊　＊

というわけでまた来月。

吾輩はX68000である

[第21回]

ちょっと寄り道,立体視

Izumi Daisuke
泉　大介

飛び出る画像
そんなコピーが懐かしい
裸眼立体視に挑戦してみる

順当にいけば，今回は前回のキー入力の仕組みを応用した小品をお届けするところなのだが，このところOh!X誌上で妙に注目されている，ある分野が気になってしかたがない。そう，裸眼立体視である。別に最近になって研究が始まったものというわけでもないし，Oh!MZ, Oh!Xを通じて何度となく取り上げられ，解説され，議論されてきたものである。にもかかわらず，この人気。人間にとって，平面に描かれたグラフィックが立体感をともなって見えるというのは，いつの時代にもかなりのインパクトを与えるものらしい。

そもそもなぜ平面に描いた図形が立体に見えるのかという議論は，最近の諸氏の原稿で説明されているので詳しくは立ち入るまい。基本的には，人間の目の錯覚(あるいは脳が高級なのか)によるのである。まず右目を閉じて，左目で吾輩のマウスを真上から見下ろしていただきたい。もちろんこのときには，マウスを上から見た図しか脳には入ってこない。そのままの状態で左目を閉じ右目を開くと，今度はマウスを右斜め上から見下ろしている図が脳に飛び込んでくる。片目で見たときに脳に入るイメージは平面のそれなので，この実験は2枚の平面図を得た以上に大した意味はない。ところがその2枚の平面図が両方の目に同時に入ると，人間の脳はそれを立体に再構成したイメージを作り上げるのだ。つまりは，ある風景なり立体なりを左目で見た図と右目で見た図を用意し，それぞれの目にそれぞれの図を入れてやれば，人間はそれを立体として認識できるわけである。

うちの御仁も以前は立体視については並々ならぬ興味を持っていたことがある。まだ吾輩が御仁のもとにやってくる前，MZ-2000君が御仁のお相手をしていた頃の話である。高速グラフィックパッケージMAGICの「自由な視点で3次元の図形を2次元に投影する」機能を使って，立体図形を左斜め前から見た2次元投影図を赤で，右斜め前から見た図を青で画面に表示して赤青メガネで遊んだわけである。このような古典的な赤青メガネ方式は，いまやすっかり廃れたものだと思っていたら，X68000用のレイトレーシングソフト「MIRAGE」や，IBM PC用のグランドキャニオンを飛び回るフライトシミュレータに採用されているのを知って驚いた。古くさくて，安値で，立体視のサンプル程度ならまだしも，本格的なアプリケーションに使うには一抹の気恥ずかしさがともなうということで敬遠されがちだが，ここまでやられるとさすがに敬服せざるを得ない。

赤青メガネ方式は，赤いセロハンが赤い光しか通さず，青いセロハンが青い光しか通さないことを利用して，左右の目に別々のイメージを送り込んでいる。しかしながら，左目に入る赤い光と右目に入る青い光からひとつの立体のイメージを作り上げるのは，さしもの人間の脳をもってしても多少やっかいなようだ。むしろ，背景をマゼンタ(赤と青の混色)にし，左目用のイメージを青で，右目用のイメージを赤で表示したほうが認識しやすいかもしれない。これなら，左右の目にはそれぞれのイメージが黒っぽい色で入るからである。おっと，セロハン方式の立体視は今回の話題とは関係ない。この話はここまでにしておこう。

◆裸眼立体視

裸眼立体視はその名のとおり，赤青メガネや液晶シャッター方式のメガネを使用することなく，人間の目だけを使って立体視を行う方法である。道具を使わないので手軽なのだが，ちゃんと立体に見るためにはそれなりの技を修得しなければならない。

技には図1のように2種類の方法がある。ひとつは平行法と呼ばれるもので，これは左側の図形を左の目で，右側の図形を右の目で眺める方法である。左の図形と右の図形の間に下敷などを立て，遠くを見るようなつもりで目の緊張を緩めると，中央に立体の図形が見えてくる。もうひとつは交差法と呼ばれるもので，左側の図形を右の目で，右側の図形を左の目で眺める方法である。図を目の前5cmくらいに近づけ徐々に寄り目にしていくと，

図1　裸眼立体視の2つの方法

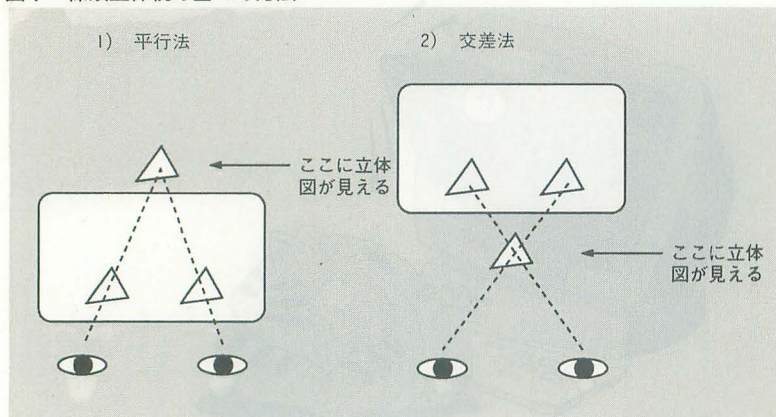



図2　平行法と交差法を見比べる

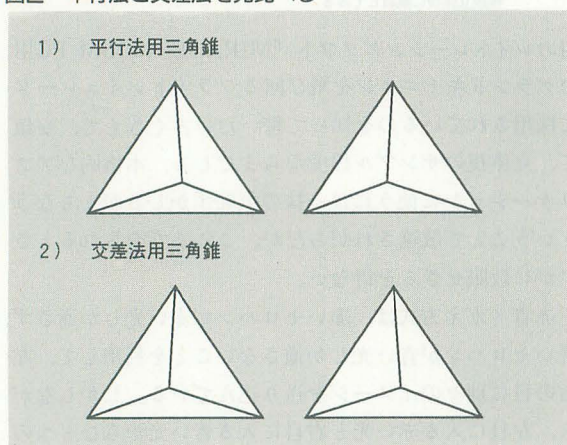
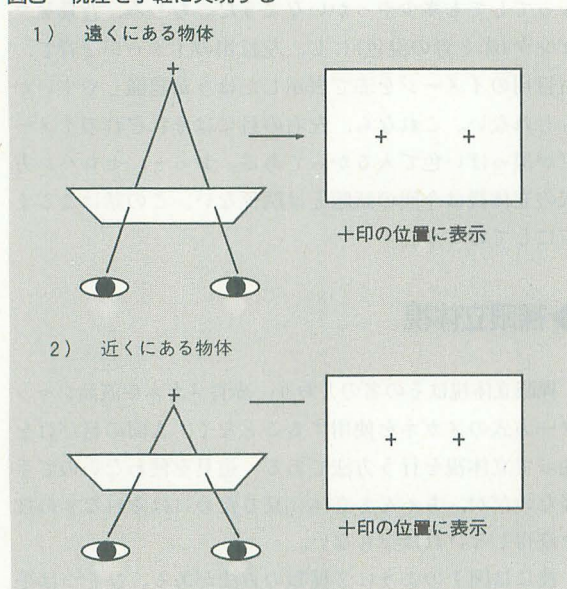



図3　視差を手軽に実現する

やはり2つの図形の間に立体の図形が見えてくる。いったん見えたら，図を目から離したほうが目への負担は少ない。

いずれにしても，前者は遠くを見るような目つきをしながら焦点をその手前の図形に合わさねばならず，後者は近くを見るような目つきをしながら焦点はその奥にある図形に合わさなければならないという，目の本来の使い方を欺くものなので，最初はなかなか立体に見るのが難しい。図1をご覧になればおわかりのように，立体視とはとどのつまり，現実には存在しない虚像を見ているにすぎないからだ。人によって平行法，交差法の得手，不得手があるようである。が，あるときちょっとしたきっかけで見えるようになると，以後は，なぜいままでできなかったのかと不思議に思うほど簡単に立体視できるようになる。

Oh!Xの今回の流行は，平行法立体視が主流になっているように見受けられるが，平行法用の図形を交差法で眺めることも，あるいはその逆も可能である。ただしその場合は，図形が裏返しになってしまう。図2に簡単な平行法用，交差法用の図を用意したので実験してみていただきたい。いずれも真ん中の頂点が手前にくるように作成してあるのだが，平行法，交差法を逆にすると，この頂点が奥にあるように見えるはずである。

◆お手軽立体視

3次元の物体を左目用，右目用のイメージとしてディスプレイに投影するには簡単な座標変換をし，陰面処理をする必要があるが，手軽に立体視を楽しみたいという程度ならばなにもそこまで本格的にやる必要はない。2次元の物体を奥行をもたせて並べるだけでも，立体視の世界を存分に楽しむことができる。では，どうすれば2次元の物体を奥行をもたせて並べることができるかだ。簡単な例を挙げてみよう。図3である。

ここでは“＋”を遠くに表示する場合と近くに表示する場合を取り上げてある。遠くに表示する場合は，両方の目の視線は画面の両端付近を通り，近くに表示する場合はそれより中央に寄ったところを通っているのを確認していただけるだろうか。図の右にはこのときの画面表示を付加してある。遠くに表示する場合と近くに表示する場合の違いは，画面に表示される左目用の“＋”と右目用の“＋”の間隔の差でしかない。単にそれだけである(この図は平行法用。交差法では遠くにあるものは間隔を狭く，近くにあるものは間隔を広く表示しなければならない)。

にわかには信じ難いかもしれないが，所詮人間の目などというものはその程度である。リスト1に簡単なX-BASICのプログラムを用意したので実行して確認してみていただきたい。実行すると画面に2つの“＋”が出現する。以後何かキーを押すたびに，両者は間隔を狭めながら下に移動していく。立体視しながら実行してみていただきたい。“＋”がしだいに手前へ向かってくるのが(交差法では遠ざかっていくのが)確認できるだろう。近寄ってきても“＋”の大きさが変わらないので，ちょっと確認しづらいという方は，リスト1の80，90行を，

```
80  draw(264+y/5,1)
90  draw(504-y/5,1)
```

と書き換えてみるとよかろう。“＋”が灰色の影を残しながら近づいてくるので，より立体感が増すはずである。

◆3Dでお絵描きを

同じ図形を間隔を違えて表示するだけで立体視用のデータとなる，ということを利用してちょっと面白いプログラムを用意してみた。立体視しながら絵が描けるグラフィックツールである。いわば3Dのグラフィックエディタといったところだが，機能的にはそんなに大したものではない。できることは自由曲線の描画とカーソルの前後移動だけ。ライン描画もなければボックスやボックスフルもない。それでも，かなり楽しんでいただけることを期待している。

プログラムはリスト2である。マウスを使ったお絵描きのプログラムは以前にもお届けしたことがあるので，ここではそれ以外の部分について補足しておきたい。

まずレジスタの用途だが，D6.Lは左目用マウスカーソルと右目用マウスカーソルの間隔を，D7.Lはマウスカーソルの座標を保持するのに使用している。

そのマウスカーソルは，mscur_off，mscur_onとラベルをつけた部分で独自に作成，表示している。描画中のグラフィックとの重なり処理を簡単にするため，IOCSコール$1B_H$を使って8×7ドットの“＋”型をテキスト画面に表示しているのがそれである。

マウスカーソルを手前や奥に動かすにはキーボードを使用することにした。“a”キーで奥に，“z”キーで手前に移動する。これはdecz，inczとラベルをつけた部分で処理している。先のリスト1では，“＋”印が手前に動くときには左目用右目用の両方の“＋”印を画面の中央に向かって移動させたのだが，ここでは右目用のカーソルを左目用カーソルに近づけるにとどめている。このため，リスト1方式では両方の“＋”印は一気に2ドット近づくのに対し，リスト2では1ドットずつ近づけることが可能となっている。

まぁ，概略はこんなところだろうか。あとはリスト中のコメントを参考にしながらプログラムを読んでみていただきたい。

最後に3Dお絵描きツールの使い方について触れておこう。プログラムを実行すると，画面にシアンの“＋”印が2つ現れる。まずはこれを立体視していただきたい。立体視が完了したら，マウスの左ボタンを押しながらマウスを移動させる。これで画面に線が表示されるはずである。「全然立体に見えない」などと早まらないように。立体視はこれからが本番だ。“z”キーをチョンチョンと5，6回押してみていただきたい。いかがだろうか。マウスカーソルが先ほど引いた線の手前に浮き上がってくるのを(交差法で立体視している方は線の奥に沈んでいくのを)確認していただけるだろう。ここで線を引くと，見事に先に引いた線と立体交差する線が完成する。もうあとは想像力次第。手前と奥の2本の線の間にもう1本線を引くなどして遊んでいただきたい。ESCキーを押せば，3Dお絵描きツールは終了である。マウスカーソルはテキスト画面に表示してあるので，

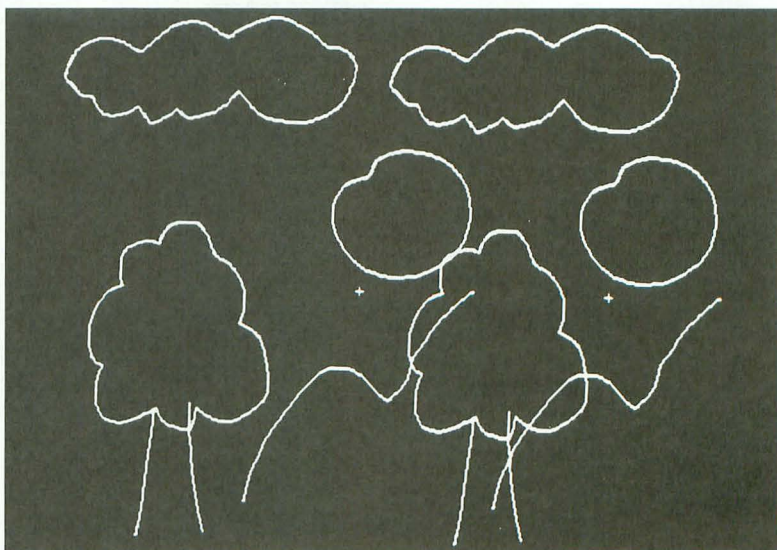
A>CLS

で消去できる。

ところで，このツールで奥から手前へと伸びる線は引けないのだろうか。いやいや，少々面倒だが可能である。要するに，線を引きながらところどころで“z”キーを押せばいいのだ。この方法で引いた直線は，実際には階段状の線なのだが，見た目にはそれほど違和感はない。奥から手前に向かっていくつか円を描いておき，それを串刺しにするラインを引く。平行線を描いておいて，それを手前へ奥へとよけながら波線を引くなど，楽しい遊びを満喫していただきたい。

また，ところどころで“z”キーを押す手順を自動化する，すなわち，ラインを描いたら自動的にカーソルがひとつ手前に移動するようにプログラムすれば，奥から手前へと伸びてくるラインを描きやすくなる。具体的には，mscur1とラベルをつけた行の前に，inczを呼び出

リスト1

```
 10 screen 2,0,1,1
 20 str a
 30 y=100
 40 draw(264+y/5,15)
 50 draw(504-y/5,15)
 60 while y<150
 70   a = inkeys
 80   draw(264+y/5,0)
 90   draw(504-y/5,0)
100   y = y + 1
110   draw(264+y/5,15)
120   draw(504-y/5,15)
130 endwhile
140 end
150 /*
160 func draw( x, p )
170   line(x,y+5,x+10,y+5,p)
180   line(x+5,y,x+5,y+10,p)
190 endfunc
```

3D.Xサンプル写真

すプログラムを挿入すればいいわけだ。毎回inczを呼び出したのではアッという間にカーソルが手前にきてしまうので，適当なインターバルを設ける必要はあるが。この部分は諸兄に改造していただこうと，あえて今回のプログラムには含めなかった。自動的にせりあがってくるラインはかなりインパクトがあるので，是非とも挑戦してみていただきたい。また，ラインやボックスなどの描画機能の追加，カラー対応などの機能の追加にも挑戦して，実用になる3Dお絵描きツールを作成していただけると幸いである。

リスト2

```
-z0=200000
-an .z0
        †_exit          equ     $ff00
        †_conctrl       equ     $ff23

00200000        move.w  #3,-(sp)
00200004        move.w  #14,-(sp)
        †       dc.w    _conctrl            * ファンクションキー OFF
00200008        _conctrl
0020000A        addq.l  #4,sp
0020000C        move.w  #16,d1              * 768×512ドット×16色
00200010        moveq   #$10,d0             * _crtmod
00200012        trap    #15
00200014        moveq   #$90,d0             * _g_clr_on
00200016        trap    #15
00200018        moveq   #$af,d0             * _os_curof
0020001A        trap    #15

0020001C        move.l  #$01080100,d7       * (264,256) : マウス座標初期値
00200022        move.l  #240,d6             * 左右オフセット初期値

00200028        moveq   #$70,d0             * _ms_init
0020002A        trap    #15
0020002C        moveq   #0,d1               * マウス右ボタン解放
0020002E        moveq   #$7d,d0             * _skey_mod
00200030        trap    #15
00200032        move.l  d7,d1               * マウスカーソル座標セット
00200034        moveq   #$76,d0             * _ms_curst
00200036        trap    #15
        †       bsr     mscur_on            * マウスカーソル表示
00200038        bsr     .z0+$106

        loop:
0020003C        moveq   #$01,d0             * _b_keysns
0020003E        trap    #15
00200040        tst.l   d0                  * キー入力があったか
        †       beq     poscheck            * なければ次の処理へ
00200042        beq     .z0+$70
00200046        moveq   #$00,d0             * _b_keyinp
00200048        trap    #15
0020004A        cmpi.b  #27,d0              * 押されたキーは ESC か
        †       beq     exit                * ESCなら終了
0020004E        beq     .z0+$82
00200052        cmpi.b  #'a',d0             * 押されたキーは a か
        †       bne     loop1               * 違うならloop1へ
00200056        bne     .z0+$5e
        †       bsr     incz                * z座標を大きく
0020005A        bsr     .z0+$ae
        loop1:
0020005E        cmpi.b  #'z',d0             * 押されたキーは z か
        †       bne     loop2               * 違うならloop2へ
00200062        bne     .z0+$6a
        †       bsr     decz                * z座標を小さく
00200066        bsr     .z0+$9c
        loop2:
0020006A        moveq   #0,d1               * キーボードLEDをすべて消燈
0020006C        moveq   #$03,d0             * _key_init
0020006E        trap    #15                 * これでキーバッファもクリアされる

        poscheck:
00200070        moveq   #$75,d0             * _ms_curgt
00200072        trap    #15
00200074        cmp.l   d0,d7               * 前回のマウス座標と比較
        †       beq     poscheck1           * 同じなら再びループ
00200076        beq     .z0+$7e
        †       bsr     mscur               * 違うなら マウスカーソル再描画
0020007A        bsr     .z0+$c2
        poscheck1:
        †       bra     loop
0020007E        bra     .z0+$3c

        exit:
00200082        move.l  #-1,d1              * マウス右ボタン制御 ON
00200088        moveq   #$7d,d0             * _skey_mod
0020008A        trap    #15
0020008C        moveq   #$ae,d0             * _os_curon
0020008E        trap    #15
00200090        clr.w   -(sp)
00200092        move.w  #14,-(sp)
        †       dc.w    _conctrl            * ファンクションキー ON
00200096        _conctrl
00200098        addq.l  #4,sp
        †       dc.w    _exit
0020009A        _exit

        *
        *   マウスカーソルのz座標を変更する
        *
        decz:
0020009C        tst.b   d6                  * D6をチェック
        †       beq     decz1               * 0なら終了
0020009E        beq     .z0+$ac
        †       bsr     mscur_off           * マウスカーソル OFF
002000A2        bsr     .z0+$fc
002000A6        subq.b  #1,d6               * オフセットを1つ小さく
        †       bsr     mscur_on            * マウスカーソル ON
002000A8        bsr     .z0+$106
        decz1:
002000AC        rts

        incz:
002000AE        cmpi.b  #240,d6             * D6をチェック
        †       beq     incz1               * 240なら終了
002000B2        beq     .z0+$c0
        †       bsr     mscur_off
002000B6        bsr     .z0+$fc
002000BA        addq.b  #1,d6               * オフセットを1つ大きく
        †       bsr     mscur_on
002000BC        bsr     .z0+$106
        incz1:
002000C0        rts

        *
        *   マウスカーソルの位置更新と
        *   ラインの描画
        *
        mscur:
        †       move.l  d7,linedata         * ラインの始点座標セット
002000C2        move.l  d7,.z0+$140
        †       bsr     mscur_off           * 以前のカーソル消去
002000C8        bsr     .z0+$fc
002000CC        moveq   #$75,d0             * _ms_curgt
002000CE        trap    #15
        †       move.l  d0,linedata+4       * ラインの終点座標セット
002000D0        move.l  d0,.z0+$140+4
002000D6        move.l  d0,d7               * マウス座標保存
        †       bsr     mscur_on
002000D8        bsr     .z0+$106
002000DC        moveq   #$74,d0             * _ms_getdt
002000DE        trap    #15                 * D0 : $XX_YY_LL_RR
002000E0        tst.w   d0
        †       bpl     mscur1              * 左ボタンが押されていなければ終了
002000E2        bpl     .z0+$fa

        †       lea     linedata,a1
002000E6        lea     .z0+$140,a1
002000EC        moveq   #$b8,d0             * _line
002000EE        trap    #15
002000F0        add.w   d6,(a1)             * 始点x座標を補正
002000F2        add.w   d6,4(a1)            * 終点x座標を補正
002000F6        moveq   #$b8,d0             * _line
002000F8        trap    #15
        mscur1:
002000FA        rts

        *
        *   マウスカーソル消去・表示
        *
        mscur_off:
        †       lea     curoff,a1
002000FC        lea     .z0+$122,a1
        †       bra     mscur_on1
00200102        bra     .z0+$10c

        mscur_on:
        †       lea     curon,a1
00200106        lea     .z0+$12e,a1
        mscur_on1:
0020010C        move.w  d7,d2               * D2.w : y座標
0020010E        move.l  d7,d3
00200110        swap.w  d3                  * D3.w : x座標
00200112        move.w  d3,d1
00200114        moveq   #$1b,d0             * _textput
00200116        trap    #15
00200118        move.w  d3,d1
0020011A        add.w   d6,d1               * D1 : x + ofst
0020011C        moveq   #$1b,d0             * _textput
0020011E        trap    #15
00200120        rts

        *
        *   マウスカーソルのデータ
        *
        curoff:
00200122        dc.w    8,7
00200126        dc.b    0,0,0,0
0020012A        dc.b    0,0,0,0

        curon:
0020012E        dc.w    8,7
00200132        dc.b    _00010000,_00010000
00200134        dc.b    _00010000,_11111110
00200136        dc.b    _00010000,_00010000
00200138        dc.b    _00010000
        †       dc.w    8,7
        †       dc.b    %00010000
        †       dc.b    %00010000
        †       dc.b    %00010000
        †       dc.b    %11111110
        †       dc.b    %00010000
        †       dc.b    %00010000
        †       dc.b    %00010000
```

[特集]

X68第7世代へ

1986年。混沌とした8ビット時代を抜け，世界が秩序に向かおうとしていたとき，ひとつの実験が行われた。
あらゆるマシンと互換性を断ち，もっとも重要といわれるソフトウェア資産を捨てて，1本のアプリケーションもない状態でパソコンを発売する……。
それは，かつてあったどんなものにも似ていないマシンだった。むしろ，ユーザーが漠然と描く夢に出てきたマシンに似ている。
発売前にはアプリがないと笑われる。
発売後まもなく，台頭する286ユーザーからCPUパワーがないと嘲笑される。
そして286は消え68000は残った。386は来年までもつだろうか。
ハードウェアを操るノウハウすらない状態から，パソコン文化を創り上げることをユーザーは強いられた。しかも，ありきたりのものでは満足しない。別に仕事で使うわけではない。だから妥協する必要はない。
そして待望の32ビットマシンが高い互換性とともに高性能を引っ提げてやってきた。
少なくともまったく新しいものによってX68アーキテクチャが否定されるという最悪の事態は避けられたわけだ。
7世代目。パワーアップという「単純な回答」の持つ意味は意外と大きいようだ。

[CONTENTS]

ハードウェアと周辺事情

X68030の目指す世界

Saitou Susumu 斎藤　晋

めでたく発表されたX68030！ ３月10日にマンハッタンシェイプの５インチFDD版が先行して発売となり，すでに買っちゃった人もいるかもしれません。まずはX68030の位置づけと周辺の事情を探ってみましょう。

先月号の訂正とお詫び

さっそくですが，３月号の記事に誤りがありましたので，この場で訂正させていただきます。コンパクトタイプの内蔵3.5インチFDDですが，残念ながら1.44MBフォーマットのディスクは読み書きできないそうです。期待していた人，ごめんなさい。私も仕事がら他機種を触らざるをえないので，3.5インチ版を買おうと思っていたのですが。もちろん，2DDは読み書きできますので，MacintoshやDOS/Vマシンなどとのデータ互換はなんとかなるでしょう。

ちょっと具体的に補足しておきましょう。昨年発売となったCompactでは2HDのみの対応となっていましたが，これはシャープ独自のオートイジェクト回路の仕様によるものでした。今回のX68030では，この回路が見直されると同時に，IOCSでも2HD以外のメディアがサポートされました。1.44Mが読めないのは内蔵ドライブ側が対応できないためです。そもそもオートイジェクトのできるドライブを製造するメーカーが限られていることもあって，コスト的にもやむをえなかったようです。オートイジェクトでなくてよければ，ツクモのTS-3XRシリーズで1.44Mのフロッピーも読み書き可能です。

高くてなにが悪い？

はっきりいってしまえば，X68030は高いです。だからどうしたっていうんだ，ってな気にもなりますが，高いのは事実だからしかたがありません。ただ，多くのユーザーが発表された価格について現実的なものと受け止めているようです。

結局，過剰に値段の高さを問題にするのはどちらかというと他機種のユーザーです。むしろX68000ユーザーの場合は，価格が上がっても機能強化を優先してほしいという声が少なくないのです。

なぜなら，X68000はもはやユーザーにとって他機種では代替のきかないマシンとなっているからです。X68000の使いやすさ，自由度の高さはほかに比較しうるマシンがないといっても過言ではないでしょう。

X68000/X68030が高いといわれる背景には，IBM PC/AT(互換機)から発生してMacintoshにまで影響を及ぼした価格競争があるのはいうまでもありません。

何度か紹介されているように，IBM互換機が安いのは互換部品メーカーの激しい競争があったためです。いまでは，IBMが作っているのも“AT互換機”といった状態です。インテルも互換CPUへの対策として，上位CPUへの移行を早めざるをえません。486の価格が大幅に引き下げられたのもそのためです。

しかしどうでしょう。結果としてユーザーは得をしているのでしょうか。激しい価格競争と上位機種への移行によって，インテル系の機種では，ほんの１～２年で過去のマシンとなってしまいます。いくらマシンが値下がりしても，毎年のように新しい機種に乗り換えていたのでは安いといってよいかどうか疑問です。

一方，Macintoshも対抗上かなり大幅な価格引き下げを図ってきました。もともと割高感のあったMacintoshですが，価格が高かった理由はソフトウェアの開発やユーザーインタフェイスの研究に膨大なコストがかかっていたためでしょう。ハードウェア自体はIBM PCと同じくワンボードマイコンにディスプレイとキーボード，マウスをくっつけたようなものですから，高くつくわけはありません。ソフトの開発費やデザイン料は量産効果で解消できるので値段は下がって当然ともいえます。

それでも，無理なローコスト化は，それなりに問題を抱えているようです。Macユーザーに聞くと，最も安心して利用できるのは，SE/30，IIciといった数年前のマシンで，最近の安普請な機種はあまり評判がよくありません。

X68030の真の実力は

X68030は，25MHzの68EC030を採用しています。68EC030はMMUがないだけで処理能力は68030と同等であるということは３月号でも紹介したとおりです。

現在では68030といっても驚いてくれる人は少なくなりました。同じ68系のCPUを使っているMacintoshではすでに68040への移行が進んでいますし，68030はローエンド機種でも当たり前のように使われています。ある意味でMacintoshが68030のイメージを下げてしまったともいえるのですが，ClassicIIやLCIIのように動作速度も16MHz，16ビットアーキテクチャのままのハードウェアで，負荷の大きなシステムを動かしていたのでは無理もありませんね。

68030というCPUがもつ本来の能力がいかにすごいかは，きっとX68030が証明してくれるでしょう。メモリアクセスは32ビット，25MHzでほぼノーウェイト。そしてなによりもX68030ではハードの仕様とソフトの開発環境がユーザーに開かれています。X68000で育ったアセンブラプログラマがこのマシンと取り組めば，MPU68030の能力とX68030の機能をあますところなく引き出してくれるに違いありません。

X68000シリーズは機能的にも1024×1024ドットのビットマップテキスト表示，512×512ドット/65536色をはじめとするさまざまなグラフィック表示，128プレーンのスプライトと２画面のBGを独立してもち，これらを合成した最大７重画面スクロール，半透明重ね合わせ表示などといった強力な機能は他機種では考えられません(これらのコントローラを３つばかりなくすだけで

定価が10万円は安くなるといわれている)。漢字も16ドットだけでなく24ドットフォントまでROMで用意されている贅沢ぶり。ステレオFM音源にAD PCMといったサウンド機能を標準で装備。またグラフィックやAD PCMデータを高速で転送するのに必要なDMAも他機種にない強力なものが搭載されています。

こうした強力かつ豊富な機能が32ビットのX68030でも使えるというのは実に素晴らしいことではないでしょうか。

動作速度とメモリの事情

皆さんよくご存じのように,コンピュータは水晶発振器によるクロックをベースに動作します。CPUに10MHzとか25MHzとかついているのが動作クロックで,1クロックの時間はこの数値の逆数となります。CPUが同一クロック内に行える処理は,CPUによっても命令の内容によっても違います。

さて,CPUの動作速度が上がるとまず問題となるのがメモリの速度です。メインメモリには一般にDRAMが使われ,X68030で使用されているDRAMのアクセスタイムは70n秒と高速な部類に入ります。しかし,単純に計算するとDRAMのスピードはCPUには追いつきません。

アクセスタイムは,DRAM内のアドレス指定からデータを読み出すのにかかる時間ですが,読み出しはDRAMにとってはコンデンサの放電を意味します。放電したら当然充電が必要で,このプリチャージの間はメモリアクセスができません。そこで単純なメモリアクセスを行う限り120n秒(かもうちょっと)がメモリの実効速度ということになるようです。

X68030の場合,動作クロックは25MHzで,メモリアクセスは基本的に2クロックですから,$1/25 \times 10^6$秒×2クロックで80n秒ということになります。アクセスタイムは70n秒ですから,最初のアクセスには十分ですが,読み出したあとにはプリチャージの時間を要するため,連続して読み出す場合には追いつかない計算です。

実際には,DRAMのアドレス指定は,行アドレスを指定したあと列アドレスを指定することで決まります。ところが,連続したアドレスをアクセスする場合,同じ行の隣の列というケースがほとんどのはず。そこで行アドレスは指定したままで,列アドレスを続けて指定するというのが,よくあるページモードという方式です。最初のアドレス指定からメモリの内容を読み出すには通常のアクセスタイムを要しますが,次の列からメモリを読み出す時間はその3分の1以下ですむようです。また,先のプリチャージは行アドレスの指定信号が消えたあとで始まるので,メモリの実効速度もかなり稼げることになります。現在はたいていのDRAMがページモードに対応しており,制御回路も比較的簡単ということで,多くの機種で採用されているようです。

しかし,X68000シリーズでは,このページモードをさらに賢くしたスタティックカラムRAMが採用されています(そうだったのか)。これは列アドレスも一度指定するだけであとは自動的に書き換えてくれるという方式で,制御回路は多少複雑になるようですが,速度的にはページモードに比べても高速です。

私は,どうしてX68000はSIMMを使えるようにしないんだ! と疑問に思ってきたのですが,一般のSIMMにはスタティックカラムタイプのものがないそうです。価格の安いSIMMでは80n秒以上の遅いRAMも多数出回っており,ページモードで遅いRAMを使ったのでは25MHzで動かす意味がなくなってしまうというのが最大の理由のようです。AT互換機などではSRAMによるキャッシュを持たせることも多いのですが,キャッシュはあくまでキャッシュです。速いCPUと速いメインメモリをノーウエイトのバスで結ぶというのがシャープの考える速いマシンの基本ということでしょう。

世間を騒がすローカルバス

さて,コンピュータではデータの経路をバスと呼んでいますが,最近,ローカルバスという言葉がパソコン雑誌を賑わすようになりました。外部デバイスを接続するための拡張バスに対し,ローカルバスはシステム内の局所的なバスという意味です。が,ここで問題となっているのは“ローカルバス(Local Bus)”という名の拡張バスで,具体的にはCPU回りのローカルバスに直結した外部バスのことをいいます。

この“ローカルバス”が採用されるようになったのは,AT互換機で拡張バスが全体の足を引っ張るようになってきたからです。ATバスはIBM PCが286マシンのATとなる際に拡張された16ビットバスで動作速度は一般に8MHzでした。その後,CPUはどんどん速くなっていくわけですが,豊富な周辺機器を利用するためには拡張バスの仕様はむやみに変えられません。しかし,ATアーキテクチャではグラフィック表示を行うビデオカードなども拡張バスにつなぐようになっています。もちろんそれが功を奏してより機能の高いビデオカードが普及してきたわけですが,その半面,バスの遅さは深刻なボトルネックになっていたのです。そこでCPU回りのローカルバスから無理やり外部にバスを出し,そこにビデオカードなどをつなぐことで高速化を図ろうというのが話題の“ローカルバス”というわけです。

X68000でも拡張バスは10MHzでしたが,グラフィック関係の回路はもともとシステム内のローカルバス上にあるため,拡張バスが問題になることはありません。問題があるとすればグラフィック回路自体の機能強化は本体仕様の変更を意味しているということでしょうか。

X68000の進む道

世の中の風潮としては,コストを下げながら,見かけのスペックを重視する傾向にあります。少数の優れた技術よりも多数の安易な手法のほうが結果的に高いパフォーマンスを得られる場合もあるでしょう。たとえば,最近のPC-9800シリーズでハードディスクがIDE方式になっているのは,AT互換機市場でIDEのドライブが極端に安くなったからですが,その分大容量なものを選べば速度も向上するわけです。

そんななかで,X68030は見かけのスペックよりも本質的な“美しい機能”にこだわり続けています。市場では少数派の高級部品を使い,コンパクト化のためにあえて高価な2.5インチのSCSIハードディスクを選び,デザインに凝って高級な塗装を施すなど,あらゆる面で独自の道を歩んでいます。あくまでも美しさをもって潔しとする。それがX68030の世界なのです。

新世代システムを見る

魂(ソフトウェア)は加速する

X68030

Nakano Shuichi 中野 修一

新マシンとともに登場した新システム。名づけて「ver.3.0シリーズ」です。従来機でも使用できるというのもうれしい話ですね。ここではX68シリーズの新世代システムをまとめて見てみましょう。

さまざまな噂の渦巻く中，ついにX68030が発表されました。待望の32ビット化とはいうものの，ハードウェアの拡張はCPUのみにとどまり，あくまでもX68000シリーズの最高速モデルとして位置づけられています。従来のアプリケーションのほとんどがそのまま動作しますし，現状で問題があるものもどんどん対応していくことでしょう。そのおかげで使用感覚もほとんど変わらず，違和感はありません。一部では過剰といわれたハードウェアをさらに速いCPUでドライブすることにより，非常にバランスのとれたマシンに仕上がった感じです。正直なところ，「こういうマシン“も”ほしかった」のです。

ハードウェアの変更とあわせて，システムソフトウェアも大幅なバージョンアップがなされました。これらのソフトは従来のX68000でも使用できます。

ということで，今回の新製品の真骨頂は実はソフトウェアのバージョンアップにあるのかもしれません。編集室にもようやくマスターに近いシステムソフトウェアが届きましたので，ここで新しいソフトウェア環境がどのように変わっていくのかを解説してみましょう。

新システムHuman68k ver.3.0

まずはHuman68k ver.3.0です。

モトローラ系のCPUではOSというものはCPUの種類ごとに換えることを前提としています。これまでのHuman68kは68000CPU用に開発されていたものですので，68030で使用することはできません。しかし，今回のver.3.0は68000と68030の両方で動作するように作られたOSなのです。

X68030と同時に発表されているソフトウェアのほとんどが従来のX68000でも使用可能となっています。使えないのはCPUキャッシュ制御コマンド（ずばりCACHE.X。もちろんCASH.Xとは無関係）くらいのものでしょう。システムのベースがver.3.0へ移行することによって，68000と68030での共用プログラムの作成が容易になることが考えられます。というか，X68030対応とするためには今後のシステムはすべてver.3.0ベースに移行していきます。

ディスク高速化ドライバ群

今回のバージョンアップで新設されたドライバ(外部コマンド)にFASTIO，FASTOPEN，FASTSEEKがあります。これらがそれぞれどのような内容を持ったコマンドで，どれほどの差が出てくるのかを検証してみましょう。

●FASTIO

FASTIOはOSのファイルI/Oを高速化するためのドライバです。一度アクセスしたファイルは二度読みしない，要するにディスクキャッシュ用のドライバと考えてかまいません。基本的なところから説明しますと，キャッシュとは，一度読み込んだディスクの内容をメモリ（キャッシュバッファ）に置いておき，次にアクセスするときにはディスクを回さずバッファから持ってくるような機構のことです。

同じものを何回も読むことってあまりないと思うかもしれませんが，ディスク上のファイルをアクセスするたびに，必ずディスク上の「ディレクトリ」と「FAT」というものを読み込みます。これはディスクの内容を示すもので，この部分だけでもメモリ上に置けばファイルアクセスは高速化されます。

これは従来のCONFIG.SYSで指定していた“BUFFERS”と同じ機能ですが，従来のBUFFERSが99セクタ分までしか設定できなかったのに対し容量と機能を拡張したものととらえることができます。

キャッシュを制御するパラメータによって単にこれまで読み込んだものを読み出す以上の動作も行います。

新システムの起動画面

たとえば，先読みするセクタ数を設定できます。ディスクには連続的に記録されることが多いので，まとめて読み込んでおけば有利になることがあります。はずれた場合は遅くなるだけですので極端に大きな値を入れても無駄でしょう。

遅延書き込みモードもサポートされています。高速化コマンドで遅延？　という素朴な疑問を抱く方もいるかもしれませんが，ここでいう遅延とは，ディスクへの書き込みが遅くなるということではなく，ディスクへの書き込みを遅らせるという意味です。具体的にいうと，プログラムがデータの書き込みを連続して行う場合など，キャッシュの中身だけを変更し，ある程度揃ったところでまとめて書き込むようにするわけです。I/O機器のアクセスというのはCPUにとっては非常に待ち時間の多い処理ですから，このようにすることでかなりの高速化が期待できることになります。

しかし，書いたふりをしていても実際にはキャッシュの中身しか書き換えていないわけですから，万一，このドライバの範囲外からディスクアクセスがあった場合などは破綻が生じる可能性もあります（ほとんどないとは思いますが）。

これらのキャッシュは基本的にDOSコールに対して作用しますので，DOSを介さ

辞書メンテツール

ない操作，IOCSコールを使ったりした（さらにいえばFDCを直接操作する）場合などはキャッシュの内容とディスクの内容で不整合が出てくる可能性もあります。そこで，IOCSコールを使ったディスクアクセスをエラーとしたり，IOCSによるアクセスが行われたときにはバッファの内容を破棄するようなオプションも設定可能です。

●FASTOPEN/FASTSEEK

FASTIOはディスクのデータに対する汎用的なキャッシュシステムですが，ディスクアクセスで特に重要となるディレクトリとFATの管理をメモリ上で行おうというのがFASTOPEN/FASTSEEKです。

前述のFASTIOにはそれぞれ「ディレクトリ優先」「FAT優先」という指定もできるのですが，それでもキャッシュバッファが一杯になると，それまでのバッファが破棄されたり，キャッシュに入らなくなったりしますので，専用のドライバでディレクトリとFATだけはちゃんと管理しておこうということでしょう。

強制イジェクトをしてディスクを入れ換えるといった場合には必ずキャッシュの内容を破棄する必要があります（キャッシュフラッシュはCTRL＋F5キーで実行）。

新しい日本語環境

今回のバージョンアップのなかでもっとも大きな変更が加えられたのがASK68K ver.3.0（以下，ASK3）です。確かに変換効率はかなり上がっています。なによりも感動的なのは「賢くなった」という点よりも「馬鹿でなくなった」という点かもしれません。以前は，ときとしてほとんど理解を超えた挙動をしていたASK68Kがリーズナブルな動作を始めたのです。

使用感としては，従来のものと差し替えて文書を打ち込んでいても違和感がない，というか変な変換が少ない分だけ快適な日本語入力環境が得られます。こういう表現では従来の環境からあまり進歩がないようにも思えますが，もともと，日本語変換フロントエンドプロセッサというものはそれぞれの癖があり，ある程度辞書を鍛えて初めて使いものになるものです。ですから，新しいASK68Kは自分なりに最適化した辞書での，鍛えに鍛えてきた成果をすでに凌駕しているといい換えることもできます。

辞書やシステムは小さく，変換効率は格段に高く，変換速度も向上しているという理想的な仕上がりを見せています。ときおり，ASK68Kだなあと思わせる癖も残ってはいますが，これはご愛敬といったところでしょう。

変換効率は向上していますが，辞書の登録内容自体はそれほど変わっているわけではないようです。むしろ品詞情報や内部処理の効率化で高度な変換を行っているのでしょう。ASK3では一生のうちに使うことがないだろう語がほぼ削られ，流行の単語が追加されているようです。まあ「カノッサ」なんて一発で変換されてもあまりありがたくないかもしれませんが。もちろん「毒」も「罠」も大丈夫です。しかし，相変わらず，JIS第2水準漢字が交ざったり，ちょっと日常的でない単語，

罵詈雑言
魑魅魍魎の跋扈する
不倶戴天の敵
補間する

といったものは自前で登録するしかないようです。なお，登録語数はざっと6万語くらいだと思われます。郵便番号辞書が辞書本体から分離されています。こういったものは必要に応じて組み込めということなのでしょう。

●辞書データのコンバート

確かに，従来のASK68Kはまったく鍛えていない辞書や間違った鍛え方をした辞書では異様にストレスを感じるような変換しかしませんでした。でも，ちゃんと使っている人なら，特殊な用語などを登録してかなり鍛えた辞書を使っている方が多いでしょうから，実際の使用状況から見た変換効率はそれほど向上しないかもしれません。

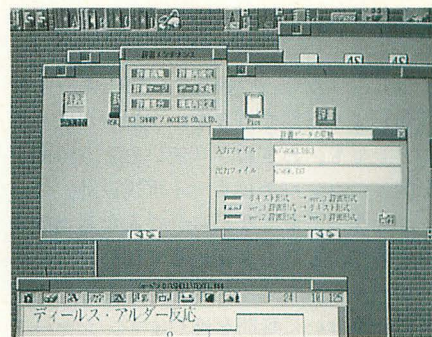
ここで，当然クローズアップされるのが従来の辞書データからのコンバートです。ASK3では辞書構造が変わっているようですから，従来の辞書は使用できません。そこでコンバータが必要になるのです。

ASK3の支援プログラムとして，SX-WINDOW用に従来の辞書からのコンバータが付属しています。

主なところでは，

1）　これまでの辞書をテキスト化する
2）　ASK3の辞書をテキスト化する
3）　辞書の差分をテキスト化する
4）　辞書をマージする

といった機能があります。基本的にテキストファイルを編集することで辞書のメンテナンスをすればいいようなシステムになっていますので扱いは楽にできそうです。

しかし，以前とほとんど変わらない辞書でこれだけの使い勝手があるのですから，これまで鍛えてきた辞書の内容を加えることで，かなり理想的な日本語環境が構築できると期待されます。

●キー設定の変更？

変換時のキー割り当てなどがユーザー定義できるのはASK2からの伝統ですがASK3では多少変更点があります。まず，入力モードではDEFINSが無視され常にインサートモードになります。DEFCONTも無視され，常に確定不要の連続入力モードとなりました。これらは特に操作性の向上につながるというものではないのですが，不便なものを取り払った感じです。また，全角/半角切り替えと平仮名/片仮名変換がそれぞれ独立のキーで設定されるようなモードが追加されています。これまで全角/半角切り替えのような操作しか認められていなかったのですが，これからはキー一発で変換ができます。

これはいいのですが，これらを同じキーに割り当てた場合，従来と同じトグル動作（押すたびに切り替わる）ができるはずなのですが，現在手元にある開発途中バージョンではこの部分がバグっているようです。巷ではX68030の発売までには間に合わないんじゃないかという憶測も有力ですが，そのうち改善されることでしょう。実は今回の原稿は期待の「シャーペン.X」で書こうと思っていたのですが，結局，この点の不備のためにSX-WINDOWに乗り換えるのをやめてしまいました。なんとWP.Xでなら昔のままのキー操作で賢い変換だけを使うことができるのです。

さらに気になるのは，標準となるべきASK1.ENVで記述されているキーアサインがこれまでのASK68Kのものと異なる点です。新しく追加されたものを利用しているだけならまだいいのですが，従来のSHIFTキーを使用した制御が意味もなくCTRLキー関係に移行されていたりします。そのままで使うと，X68000を使ったことのある人には非常に使いにくい設定といえます。

もちろん，従来の環境設定ファイルがほぼそのまま使えるので実使用上は問題はないのですが，6年間にわたって守られてき

た操作系の互換性がここで破綻しているのです。

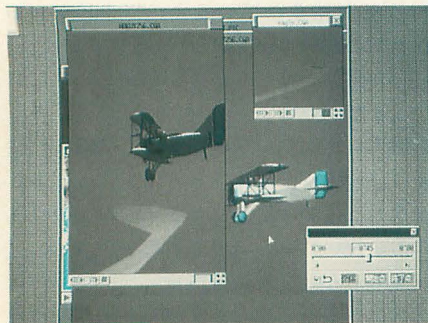
仮に，こういった操作にCTRLキーを使うのがより自然であるという解釈が成り立つとしても，わざわざ「使いやすさの向上を図ったコンパチビリティ重視設計」のマシンを作っておきながら，こんな基本的なところで，もっとも変更してはいけないユーザーインタフェイスを変更するという姿勢にはおおいに疑問を感じます。シャープはいったいなにを考えているのでしょうか？

●アクセサリ機能

ASK3では日本語入力時に独自のアクセサリツールを呼び出せることになっています。現状では，具体的にどのような内容のものが提供されるのかわかりませんが，文字入力が行えるところならどこでも使える付加機能，たとえば，現在の電卓機能が強化されたものと考えればわかりやすいでしょうか（あるいはStationeryPRO-68K, TeleportionPRO-68Kのようなもの）。

語句の変換中にその機能の起動キーを押します。すると，現在変換中の文字列を引数として，なにかの機能が働くわけです。その「なにか」に当たるものはなんでもいいので，「その語を見出しとしてデータベースを検索する」「漢字の部首別辞書を検索する」「数式とみなして関数電卓を呼ぶ」「WORDPOWERなどのような特殊な辞書を呼ぶ」などが考えられます。これらのアクセサリの実体はあらかじめ登録しておいた常駐プログラムです。

アクセサリ開発に必要な機能は日本語FPコールの追加というかたちで提供されます。関連する資料は公開されますので，ユーザーレベルでもアクセサリの開発ができるようになります。

基本的に日本語変換ウィンドウと候補ウィンドウを使うものなのである程度アプリケーションは限られてきます。画面を壊さないように留意すればゲームを起動するといったこともできなくはないでしょうが，

アニメーションも再現できる

安全を期するなら変換ウィンドウ内でできる電卓インベーダ程度にとどめておくべきでしょう（ああポケコン文化！）。

こういったメーカー標準のアクセサリインタフェイスが用意されたことで，デスクトップ環境はどんどん拡張されていくことでしょう。おそらくアクセサリ集なども発売されてくるのではないかと思います。

SX-WINDOW ver.3.0

今回のソフトウェアバージョンアップの真打ちといえば，やはりSX-WINDOWです。まず全体的な高速化，さまざまなマネージャの拡張，近日予定されている開発ツールのリリース，そしてなにより，SX-WINDOWを快適に駆動できるマシンが登場したという事実により，SX-WINDOWがようやくX68000シリーズの標準環境として成立しようとしているのです。

ここでは拡張された機能をひととおり眺めてみます。

●65536色モードのサポート

最大512×512ドットのグラフィック表示エリアを設定することにより，SX-WINDOW上でグラフィックアプリケーションを動作させることができるようになりました。これら65536色の表示（と変換）を一手に束ねているのがビデオマネージャIVM.Xです。

グラフィック表示ツールのキャンバス.Xはキャンバス.X（半角文字）と名前を変えています。そこでサポートされている画像フォーマットは，

PIX
APIC
PIC（DōGA）
TIFF（モトローラ）
TIFF（インテル）
JPEG
GL0
GL3
GLM（大きさ任意のGL3）
PNT（Easypaint）
RPC（ランレングス）
PAT4

となっています。

●アニメーション再生

ウィンドウ内でアニメーションを表示することができます。アニメーションデータはDōGA CGAシステムのデータをコンバートして使用します。基本的にはアニメーションファイルをディスクから読みながら順次再生します。AD PCMファイルを同期して再生することもできます。

マネージャが時間管理していますので，速いマシンなら滑らかに，速い記憶装置ならさらに滑らかに，というふうに動作します。さらにファイルから読み込まずにメモリ上にデータを置く指定もできますので，メモリがあればそれだけ滑らかな再生が可能です。アニメーションさせる画面の大きさも可変ですから，表示範囲を小さくすればより滑らかな再生ができます。と，それぞれの環境に合わせて適切な対応がとれる柔軟なシステムです。

再生速度は68030なら128×128ドットの画像を秒間30フレーム表示可能といわれています。再生できる最大ウィンドウサイズはもちろん512×512ドットです。他機種で実現しているウィンドウシステムその他のアニメーションシステムと比べると画面書き換え時のちらつきも少なく，けっこう高性能な部類に入るでしょう。

しかしデータの圧縮などでまだ疑問点も多く，実用度はいまひとつといったところでしょうか。10MHzマシンで256×256ドットのデータをフロッピーから読みながら再生する場合で秒間1.5フレーム程度の再生速度となります。

アニメーションファイルは圧縮されていますが，フロッピーディスク１枚に入るデータはだいたい５秒分くらいです。アニメーションデータの圧縮はIVMのデフォルトで行われるので，RPCで効率の悪すぎるものはベタ指定しておくとよいでしょう。

とりあえず，今後はQuickTimeムービーの再生にも対応する予定とのことです。

●フォントマネージャの高速化

アウトラインフォントの表示が若干高速化されました。また，マルチフォントにも対応しています。

●プリンタサポートの拡大

キヤノンBJ系のプリンタ（カラーを含む）に対応しました。

そのほか，ようやくレーザープリンタへの対応が行われることになりそうです。キヤノンLASER SHOT，エプソンESC-Page，アップルNTX-J（PostScript）系列のレーザープリンタのサポートが予定されているようです（純正レーザープリンタは発売されない模様）。

●表示画面状態を記憶しない

ウィンドウ位置を常に覚えておくのをやめます。このモードではウィンドウ位置などを変更しても次回に開かれる状態に影響しなくなります。

●コピー機能の強化

従来はアイコンドロップのコピーなどで無条件に上書きコピーが行われていましたが，新しいSXシェルではタイムスタンプの新しいものだけコピーするといった選択もできるようになりました。一度モードを選択すると次回からはその設定がデフォルトとして機能します。

●その他

階層化ウィンドウや，階層化メニューにも対応しました。階層化ウィンドウはグラフィックウィンドウに，階層化メニューはキャンバス．Xに採用されています。

現状では階層化ウィンドウの詳細がまだわかりませんので，単にウィンドウクリッピングを行っているだけなのか，もっと高度なサービスが用意されているのか不明です。階層化メニューは複雑な選択を行うメニューを実現できますが，メニューに偏重したユーザーインタフェイスになると操作性を損なうことも十分考えられます。

シャーペン．X

最後に，新しいSX-WINDOWを象徴するアプリケーション，シャーペン.X(以下シャーペン)について解説しましょう。

「日本語マルチフォントエディタ」という広告を見てなんでフォントエディタがついてるんだろうといぶかしんだ人はいないでしょうか？　どちらかといえば「日本語マルチフォントテキストエディタ」というのが正解です。要するにワープロですね。

システムにオマケの簡易版にしては，段組み編集ができない以外はほぼひととおりの機能を備えています。これまでX68000用に発売されていた数万円するワープロと比べても遜色はありません（控えめないい方だな)。日本語変換がインラインで行われないのが難点ですが，これはそのうち改善されるのだそうです。

25行にわたるメニューとかドットを単位とした管理とか泥くさい部分もあるのですが，多少手間をかけてやればやりたいことはほとんどできるという点で，非常に使いでのあるアプリケーションだといえます。

それではシャーペンの特徴を順に挙げていってみましょう。

●マルチフォント

まず，文字の大きさが自由に設定できます。これはエディタ.Xの頃から同じですが，シャーペンではいろいろな大きさの文字を混在することができます。文字間は1文字単位で設定でき，行間も各行で独立に指定できます。

書体倶楽部などのフォントを使用すればアウトラインフォントが使用できます。“マルチフォント”と銘打っているように，1文字単位で自由な設定が可能です。装飾や文字色も1文字単位です。

●マルチテキスト/シングルウィンドウ

2つのファイルを編集する際には，2つのウィンドウを使用することもできれば，ED.Xのようにひとつのウィンドウ内でバッファを切り替えて作業を行うこともできます（シングルウィンドウモード)。

基本的にエディタ.Xの進化型ですから，各種コントロールキーやキーボードマクロなどもそのまま使用でき，普通のテキストエディタとして立ち上げることも可能です。新しいSX-WINDOWではファイルビュア，テキストエディタ，ワープロの機能がすべてシャーペンでまかなわれています。起動時のオプションで指定することで，さまざまな用途に適した起動の仕方をします。

●グラフィックの張り込み1

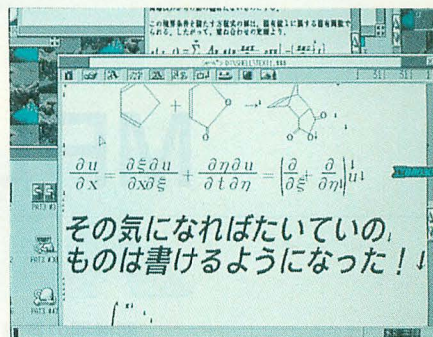
さらにシャーペンでは文書内にグラフィックを張り付けることができます。これはPAT4(アイコンなどと同じ)などのデータを使用するもので，張り込まれたグラフィックはその大きさの文字と同じ扱いになります。

応用例として，写真を見ればわかるように，やりようによっては数式，化学式，構造式などが記述可能です。最近は楽譜の記述もできそうだと考えていますが……。

さて，図版を使った文書が作成可能だといっても，シャーペン自体には作図機能はありません。そこはマルチウィンドウ環境ですから，作図専用のソフトで作成したものを張り込んでやればいいのです。で，作図専用ソフトというのは……，現状ではパターンエディタ.Xということになります。作図ウィンドウが小さいのが難点ですが，使ってみるとけっこう使えるものです。なお，あまり知られていないことですが，パターンエディタはオプションに－Fをつけることにより，メモリの許す限りの大きさの絵をエディット可能になります。

とはいうものの，パターンエディタではやはり機能的な限界や，ウィンドウの大きさの点でかなり不満は出てきます。一刻も早くテキスト画面用のグラフィックエディタが登場することを祈りましょう。

●グラフィックの張り込み2

PAT4だけではなく，PICやJPEGの画像も文書に張り込むことができます。この画像はあらかじめキャンバス.Xでコピーしておく必要があります。

多機能なシャーペン．X

この場合，画面上にはテキスト画面に変換された画像が表示されますが，カラープリンタで出力するとフルカラーで印字してくれます。

＊　　＊　　＊

マルチウィンドウ環境，そしてASK3との組み合わせにより，シャーペンの利用価値はさらに向上しています。これだけの表現力を持って，しかも高速です。

このようにSX-WINDOWの表現力も非常に向上しました。シャーペンは新しくなったテキストマンとフォントマネージャ，ビデオマネージャなどの力を結集したものです。これ程度のことが簡単にできるくらいのポテンシャルを持ったウィンドウシステムなら今後のアプリケーションの展開にも期待できそうですね。今後発売が予定されているEG Wordはさらに本格的なワープロ機能を備えたものとなります。

SX-WINDOWがこれからの標準プラットホームになっていくのは間違いないでしょう。そのためには空き容量のあるハードディスクと6Mバイト程度のメモリは必須となってきます。他機種と比べれば要求される資源は少ないのですが，快適なレスポンスを確保するにはそれなりのマシンパワーも必要です。そう考えるとX68030の存在意義もSX-WINDOWによるところが大きいといえるでしょう。ゲームやシステム関係のツールで(プログラム的な問題なしに)特に遅いと感じるものはほとんどなく，遅いのはグラフィックとSX-WINDOWだけでしたので。

しかしX68030でSX-WINDOWを扱っていても特に速いと感じることはありません。なにもストレスを感じずに操作できるのはSX-WINDOWに自然な速さというのがこれくらいだということなのでしょう。アウトラインフォントの展開とかグラフィックの変換を除けば実に軽快な動作です。この上ならなにかやってみたくなる，X68000誕生のときから目指していた環境がようやく実現しそうだ……そんな感じです。

32ビットのアーキテクチャを探る

MPU MC68030の概要

X68030

Nakamori Akira 中森　章

新機種の登場で「内部32ビット」といわれていた68000が名実ともに32ビットCPUとして使えるようになりました。ここではその基本的なアーキテクチャを概観してみましょう。

はじめに

大方の予想どおり，32ビット版X68000であるX68030のCPUは，25MHzのMC68030（以後MCは省略して68030などと記述します）ということに落ち着きました。実際にはMMUのない68EC030ですが，命令セットなどは68030と変わりありません。

68030といえば，ひと昔前まではSUN3などのワークステーション用のCPUとして全盛期にありました。最近ではRISCチップにその座を奪われてしまいましたが，かつて仕事でSUN3/260というワークステーションの恩恵(?)を受けていた私にとって，68030というCPUは非常に愛着があります。そして，もうすぐ68030マシンが自分のものになる（すでに買う気でいる）と考えるとわくわくしてきます。

私は68030がX68030に採用されたのを知るとすぐに68030関係のマニュアルを読み漁りました。68030について大体のところは知っていましたが，詳細は知らなかったので，X68030を買うまでの準備運動というところです。故人曰く，備えあれば憂いなし，ですね。そこで，それらのマニュアルから得た知識をもとに，少しばかりの想像を踏まえ，以下では68030の概要について説明していきたいと思います。

68000ファミリと68030

68030について説明する前に，モトローラの68000ファミリの開発経緯を眺めておきましょう。図1がモトローラのM68000ファミリと呼ばれる製品群のうち，プロセッサの発展関係を示したものです。また，表1に代表的なプロセッサの比較を示します。

1979年にM68000ファミリの基本的アーキテクチャを実現した68000が発表されました。命令セットアーキテクチャは32ビットになっていましたが，内部演算は16ビット単位に行われるため16ビットプロセッサに分類されます。

1982年の68008は68000から外部バス幅や制御信号を減らした廉価版です。

図1　M68000ファミリ

同じ1982年に発表された68010は68000に仮想マシンと仮想記憶サポートの機能を追加しました。分岐を高速化するループモードを採用したり一部の命令を高速化しましたが，基本的には68000と大きな差異はありません。

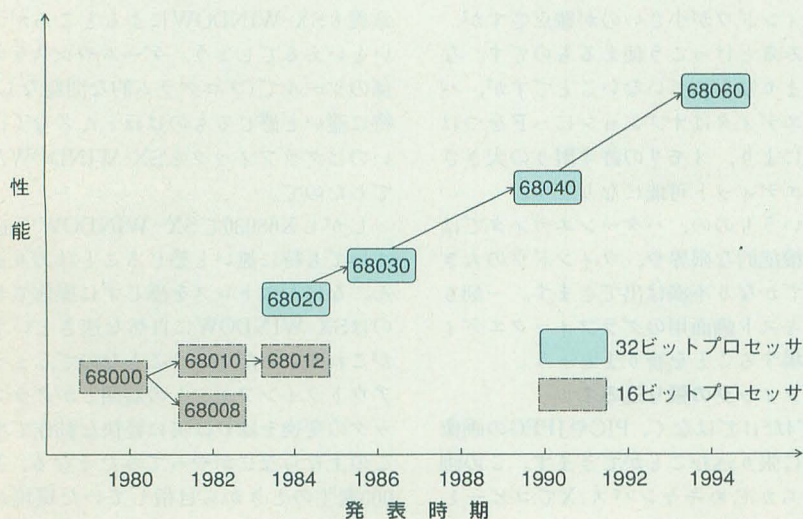
1984年の68012は68010に対してアドレスバス幅を増加したプロセッサです。中途半端なプロセッサだったせいか，現在販売はされてないようです。

1984年の68020はM68000アーキテクチャを実現する完全32ビットプロセッサです。68000の4倍の性能を実現することを目標として設計され，いろいろな高速化手法が試みられています。そのひとつが256バイトの命令キャッシュの内蔵です。68020と同時に，浮動小数点演算コプロセッサの68881とMMUの68851が開発されました。

1986年の68030は68020のマイナーチェンジです。68020に対して1.5倍の性能を実現するために256バイトのデータキャッシュを内蔵しました。また，世の中の傾向に従ってMMU（68851のサブセット）が内蔵されました。また，この年は68881の強化版である68882も開発されています。

1990年の68040は68020の4倍の性能を実現することを目標に開発されました。基本的には68030と同じアーキテクチャを採用し，キャッシュの容量増加（命令，データ，それぞれ4Kバイト），命令レベルでの高速化が行われています。

1994年（今の予定）には68060が発表されます。68060は，68040に対し，キャッシュ容量の増加（命令，データ，それぞれ8Kバイト），2つの整数演算の同時実行などによって，68040の3.5倍以上の性能を実現しようとしています。68040の次が68050でないのは，モトローラは10の位が奇数のプロセッサは前バージョンのマイナーチェンジと位置づけているためだそうです。

68000，68020，68040，68060とモトロー

ラは前の世代のプロセッサに対して約4倍の性能（動作周波数の増加を含む）を実現しようとしてきました。この世代交代の時間は約4～5年です。X68000の発売後，シャープは5年間はアーキテクチャを変更しないと宣言しました。そして，5年よりやや遅れた6年を経てX68030の登場となるわけですが，このサイクルがモトローラのプロセッサの世代交代と同期しているようで興味深いですね。

内部構造と命令実行

それでは68030の説明に入りましょう。とはいえ，X68030に採用されているのは，MMUのない68EC030ですから，ここでは主として68EC030について説明します。以後，特に断りがないときは68030とは，多くの場合，68EC030と68030の両方を指していると思って読んでください。

図2が68EC030のブロック図です。また，図3に68EC030のプログラミングモデルを示します。

68EC030の内部は大別して，68020互換な

図2　MC68EC030のブロック図

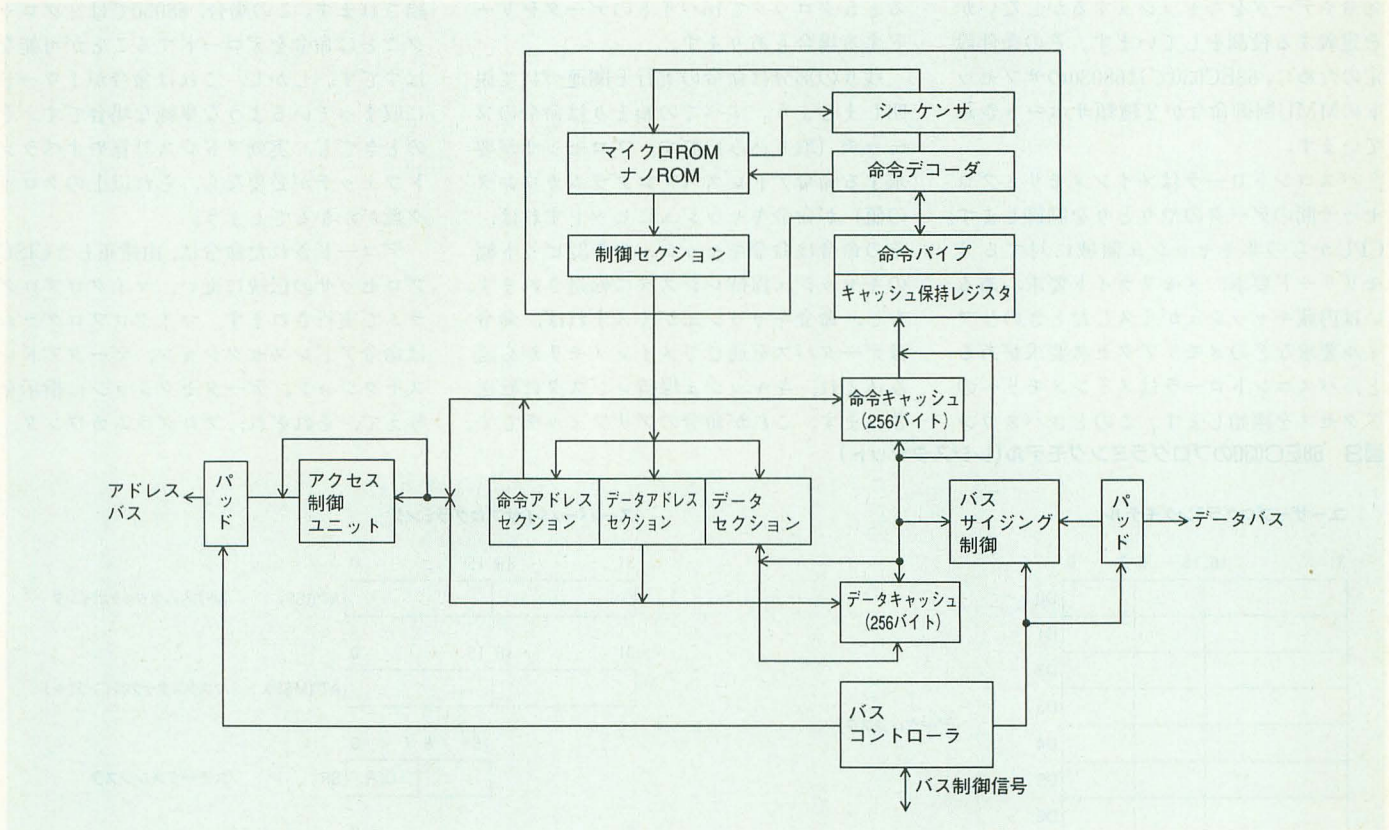



表1　代表的なM68000ファミリの比較

項目	MC68000	MC68010	MC68020	MC68030	MC68040
データバス	16ビット	16ビット	8/16/32ビット	8/16/32ビット	32ビット
アドレスバス	24ビット	24ビット	32ビット	32ビット	32ビット
命令キャッシュ	なし	なし(ループモードあり)	256バイト	256バイト	4Kバイト
データキャッシュ	なし	なし	なし	256Kバイト	4kバイト
コプロセッサインタフェイス	なし	なし	あり	あり	内蔵
仮想記憶サポート	なし	あり	あり	内蔵	内蔵
バスサイクル	非同期4クロック	非同期4クロック	非同期3クロック	非同期3クロック 同期2クロック バーストモード	同期2クロック バーストモード
ワードアラインメント制限	ワード/ロングワードデータ，スタック，命令	ワード/ロングワードデータ，スタック，命令	命令のみ	命令のみ	命令のみ
制御レジスタ	なし	ベクタベース指定 アドレス空間指定	MC68010＋ キャッシュ制御	MC68020＋ MMU制御	MC68030＋ 浮動小数点制御
スタックポインタ	USP, SSP	USP, SSP	USP, SSP(MSP, ISP)	USP, SSP(MSP, ISP)	USP, SSP(MSP, ISP)
例外スタックフレーム	オリジナル	フォーマット$0, $8	フォーマット$0, $1, $2, $9, $A, $B	フォーマット$0, $1, $2, $9, $A, $B	フォーマット$0, $1, $2, $3, [$4,] $7
アドレッシングモード	12種類	12種類	18種類	18種類	18種類
命令セット	80種類	85種類	109種類	112［109］種類	132＋29*)［107］種類

注意）　［ ］内は68EC030または68EC040の場合。
68000/68010のアドレッシングモードの数からクイックイミディエートとインプリシット参照は除いてある。
命令数はマニュアルでの項目数を数えた。基本的にはデータタイプを無視した数である。
*)はソフトウェアライブラリによる浮動小数点命令の数。

CPU部，バスコントローラ，命令キャッシュ，データキャッシュ，アクセス制御ユニットから構成されます。それでは各部分の主な動作を見ていきましょう。

アクセス制御ユニットは本来の68030でのMMUに相当します。しかし，MMUサポートがなくなった68EC030でも，その残骸が盲腸のように残っています。このユニットはあるアドレス範囲のアクセスに対して命令やデータをキャッシュするかしないかを定義する役割をしています。その条件設定のために，68EC030では68030のサブセットのMMU制御命令が2種類サポートされています。

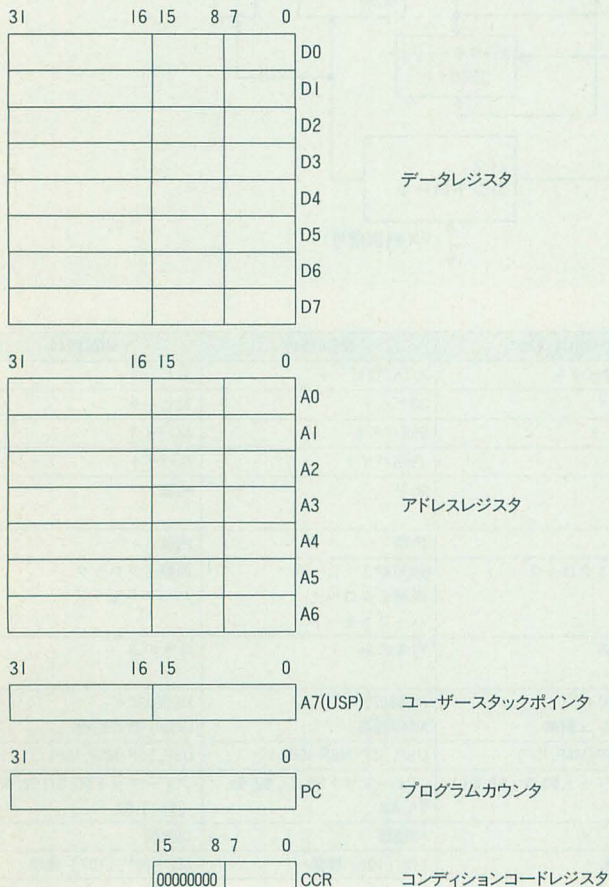
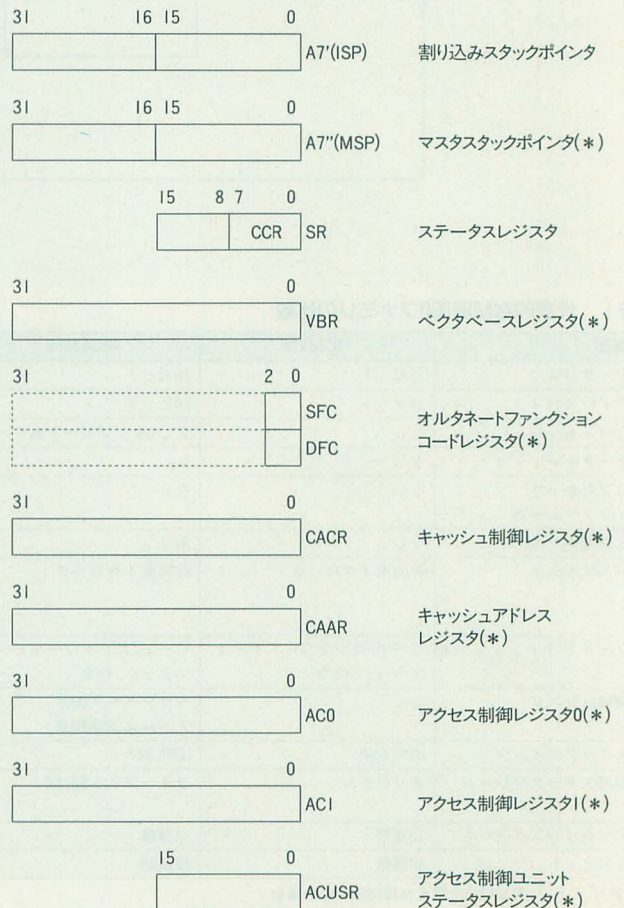
バスコントローラはメインメモリとプロセッサ間のデータのやりとりを制御します。CPUからの非キャッシュ領域に対するメモリリード要求，メモリライト要求，あるいは内蔵キャッシュがミスしたときのリフィル要求などのメモリアクセス要求があると，バスコントローラはメインメモリへのアクセスを開始します。このときバスコントローラは，バス制御信号の状況に応じて，最小3クロック，あるいは，最小2クロックのバスサイクルを起動してメインメモリにアクセスします。

そして，データバスに乗っているデータを決められたタイミングでプロセッサ内に読み込んだり，プロセッサからデータバスにデータを出力したりします。キャッシュのリフィル時にバーストモードが許可されると5クロックで16バイトのデータをリードする場合もあります。

残りの部分は命令の実行と関連づけて説明しましょう。すべての始まりは命令のフェッチ（取り込み）です。プロセッサが要求する命令アドレス（プログラムカウンタの値）が命令キャッシュにヒットすれば，その命令は命令キャッシュから32ビット幅のキャッシュ保持レジスタに転送されます。もし，命令キャッシュがミスすれば，命令はデータバスを通じてメインメモリから読み込まれ，キャッシュ保持レジスタに転送されます。これが命令のプリフェッチです。命令の実行よりも先立って（プリ＝前）フェッチが行われるためそう呼ばれます。

命令フェッチのあとはデコード（解釈）が行われます。キャッシュ保持レジスタの先には3ワードの命令パイプが接続され，そこを通過する間に命令がデコードされます。ここで，命令が命令キャッシュにヒットするような場合は，キャッシュ保持レジスタから命令パイプへ絶え間なく命令が供給されます。この場合，68030では2クロックごとに命令をデコードすることが可能なはずです。しかし，これは命令が1ワードに収まっているような単純な場合です。そのときでも，実効アドレス計算やオペランドフェッチが必要なら，それ以上のクロック数がかかるでしょう。

デコードされた命令は，由緒正しきCISCプロセッサの伝統に従い，マイクロプログラムで実行されます。マイクロプログラムは命令アドレスセクション，データアドレスセクション，データセクションに指示を与えて，それぞれ，プログラムカウンタ，

図3　68EC030のプログラミングモデル(レジスタセット)

アドレスレジスタ，データレジスタの参照や更新を行います。もし，メインメモリにあるデータが必要になれば，まずデータキャッシュを参照（直接メインメモリを読みに行く場合もある）し，キャッシュにヒットすればそこからデータを読み込みます。キャッシュにミスすればメインメモリからデータをキャッシュに読み込むためのリフィルサイクルが開始されます。

命令の実行はデコードに比べると処理が軽いので，多くの場合（加減算などの単純な処理の場合），命令実行クロックはデコード時間に依存します。つまり，命令のデコードよりも短い時間で命令を実行することはできません。68030の命令の最小実行時間が2クロックなのは命令パイプの処理単位が2クロックのためと思われます。以上が68030が命令を実行する手順です（ブロック図からの想像にすぎませんが）。このような処理を繰り返してプロセッサは動作するものなのです。

ところで，68030では，マイクロプログラム，ナノプログラムという2段階のマイクロプログラムによって命令実行がされます。ナノプログラムとはマイクロプログラムからのサブルーチンコールやシステムコールみたいなものと理解しておけばよいでしょう。こうすることで，マイクロプログラムが格納されるマイクロROMの使用効率を上げることができます。

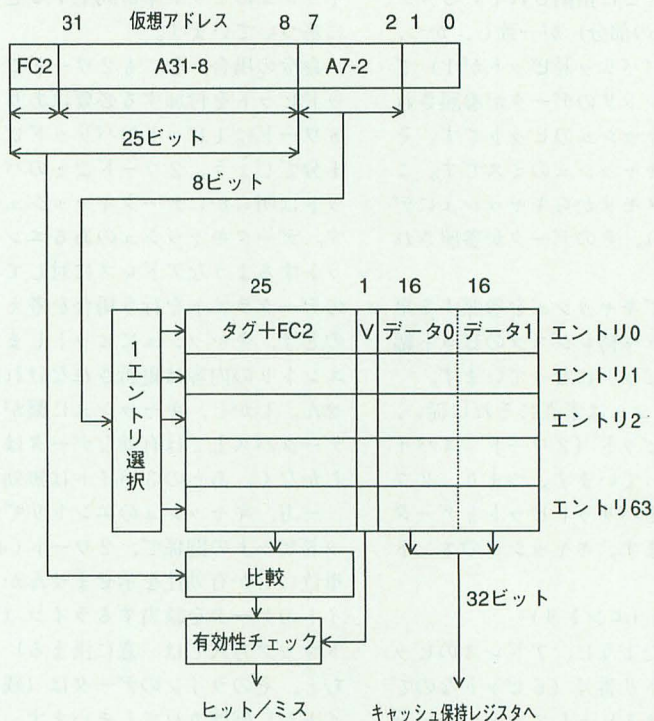
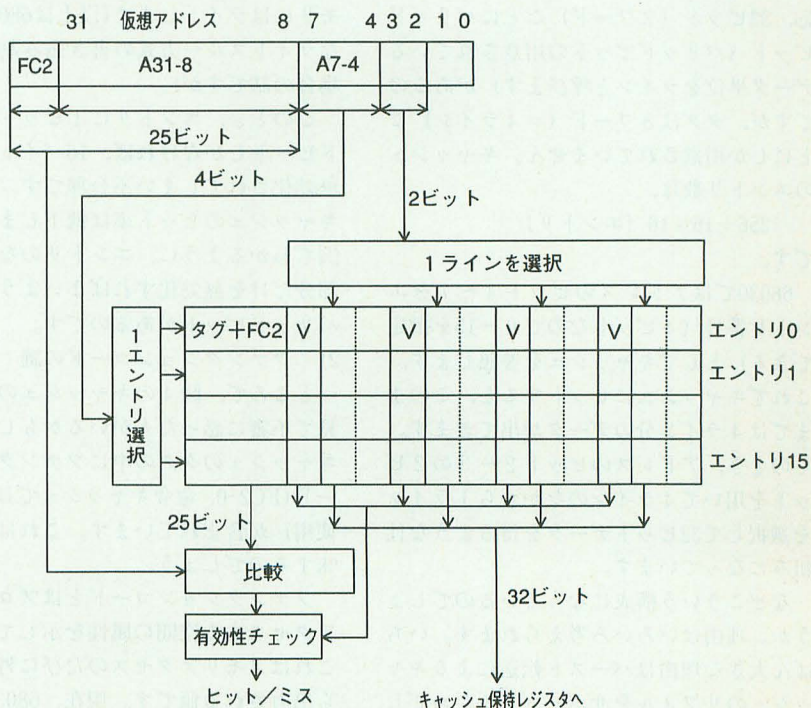
内蔵キャッシュ

1)　キャッシュの構成

68030は68020に256バイトのデータキャッシュとアクセス制御ユニット(MMU)を追加したプロセッサと理解することができます。しかし，厳密にいえば（命令セットの差異を無視したとしても），これは正しくありません。まず，バスインタフェイスは2クロックのバスサイクルや，5クロックで16バイトを転送するバースト転送が可能なように改良されています。さらに，キャッシュの構造が変わっていることも挙げられます。

図4に68020と68030のキャッシュの構成を示します。図4は命令キャッシュの例ですが，68030のデータキャッシュは命令キャッシュとほとんど同じ構成です。

68030も68020もキャッシュは256バイトのダイレクトマップ方式になっています。ダイレクトマップ方式とはあるアドレスが与えられるとそれに対応するデータが入るべきキャッシュのエントリが一意に決まる

図4　68020と68030の（命令）キャッシュ構成

V：バリッドビット
FC2：ファンクションコードのビット2

という方式です。具体的にはアドレスの一部のフィールドをキャッシュのエントリ番号とみなして，対応するキャッシュのエントリを参照し，そこに格納されているタグ（アドレスの残りの部分）が一致し，かつ，エントリが有効（バリッドビットが1）であれば，そのエントリのデータが参照されます。これがキャッシュのヒットです。そうでない場合がキャッシュのミスです。このときはメインメモリからキャッシュにデータが読み込まれ，そのデータが参照されます。

68020や68030でキャッシュを参照する単位は，キャッシュ保持レジスタのビット幅に合わせて，32ビットになっています。

68020のキャッシュは素直にそれに従い，1エントリが32ビット（2ワード＝4バイト）の構成になっています。つまり，2ワード単位にタグとバリッドビットとデータが用意されています。キャッシュのエントリ数は，

256÷4＝64（エントリ）

です。先に示したように，アドレスのビット2～7をエントリ番号（6ビットなので0～63を指定できる）としてキャッシュを参照します。

それに対して68030のキャッシュの構成は奇妙です。1エントリが128ビット（2×4ワード＝16バイト）になっています。一応，32ビット（2ワード）ごとにバリッドビット（バリッドビットの用意されているデータ単位をラインと呼びます）があるのですが，タグは8ワード（＝4ライン）ごとにしか用意されていません。キャッシュのエントリ数は，

256÷16＝16（エントリ）

です。

68030ではアドレスのビット4～7をエントリ番号（4ビットなので0～15を指定できる）としてキャッシュを参照します。これでキャッシュにヒットすると，このままでは4ライン分のデータが出てきます。そのとき，アドレスのビット2～3の2ビットを用いて4ラインのなかから1ラインを選択して32ビットデータを得るような仕組みになっています。

なぜこういう構成になっているのでしょうか。理由はいろいろ考えられます。いちばん大きな理由はバースト転送によるキャッシュのリフィルをサポートするためでしょう。1回の参照（このときひとつのタグが決定する）で16バイトのデータを高速に転送するバーストモードの処理を効率的に行うにはエントリのサイズは16バイトが最適だからです。一度にできるだけたくさんのデータを読み込むという発想は，命令のようにアドレスに連続性のある場合は，キャッシュのヒット率が向上するという事実に基づいています。

命令の場合，なにも2ワードごとにバリッドビットを付加する必要はありません。8ワードに1ビットのバリッドビットでも十分でしょう。2ワードごとのバリッドビットは明らかにデータキャッシュのためです。データキャッシュのあるエントリにヒットするようなアドレスに対してバイト長のデータライトを行う場合を考えます。このとき，キャッシュにヒットしますから，エントリの内容は更新されなければなりません。しかし，キャッシュに繋がる内部のデータバス上では有効なデータは1バイトしかなく，あとの3バイトは無効です。

一方，キャッシュのエントリでは，バリッドビットの関係で，2ワード（4バイト）単位にしか有効性を示せませんから，1バイトのデータを該当するライン（ダイレクトマップ方式では一意に決まる）に書き込むと，そのラインのデータは（残りの3バイトが）破壊されてしまいます。そこで，単純な構成のキャッシュでは，ラインのサイズに満たないデータ長のライトに対してはそのエントリを無効化するようになっています。もちろん，そのデータはメインメモリにはライトします（以上は68030のようなライトスルー方式の書き込み制御を行う場合の話ですが）。

このとき，エントリに1ビットのバリッドビットしかなければ，16バイトが同時に無効化されてしまい不合理です。明らかにキャッシュのヒット率は低下します。この例でわかるように，エントリのなかの最小部分だけを無効化すればよいように複数のバリッドビットがあるのです。

2)　ファンクションコードの謎

ところで，図4のキャッシュの構成図を見て不審に思った人がいるかもしれません。キャッシュのタグの中にファンクションコード（FC2-0，命令キャッシュではFC2のみ使用）が含まれています。これはなにを意味するのでしょう。

ファンクションコードとはプロセッサがアクセスする空間の属性を示しています。これはメモリアクセスのたびに外部端子から出力される値です。現在，68030では，

ユーザーデータ空間
ユーザープログラム空間
スーパーバイザデータ空間
スーパーバイザプログラム空間
CPU空間

の5種類が定義されています。簡単にはユーザーモードでアクセスしたアドレスかスーパーバイザモードでアクセスしたアドレスかということです。

これをタグに加えてアドレス比較を行う理由は68030が仮想記憶モードで動作することを想定しているプロセッサだからです。そして，キャッシュはこの仮想アドレスで参照され，タグ比較が行われる仮想キャッシュだからです。

一般に，ユーザーモードとスーパーバイザモードでは異なるアドレス変換テーブルが用意されています。つまり，同じ仮想アドレスを変換した結果でも，物理アドレスがユーザーモードとスーパーバイザモードで異なります。

すなわち，2つは別のデータ領域，あるいは別のプログラム領域として認識されなければなりません。割り込みやらシステムコールやらで，制御の流れはユーザーモードとスーパーバイザモードの間を頻繁に行き来しますから，仮想アドレスを見てヒットしたからといって，そのエントリのデータを参照すると誤動作することがあります。それを防ぐためにユーザーモードであるかスーパーバイザモードであるかを知るためにファンクションコードを参照しているのです。それはFC2の1ビットを見ればわかります。それなのに，データキャッシュではFC0～FC2の3ビットを参照するようです（図にはありません）。これは，未定義な空間（MOVES命令で実現できる）やCPU空間ではキャッシュを無理やりミスさせるためでしょう。

＊　　＊　　＊

以上の話は結構難しかったかもしれませんね。まあ，キャッシュの構成などは知らなくてもプログラムは書けますし，プログラムが誤動作することもないでしょう。しかし，真にプログラムの性能を引き出そうと思ったら，キャッシュの構成を考慮に入れなければなりません。プログラムの配置を最適にしてキャッシュのヒット率が最大になるような工夫が必要です。

バスサイクル

プロセッサの内部動作がいかに高速になろうと，外部とのデータのやり取りがバスネックになっていたのでは話になりません。68000ファミリは68020，68030，68040とバスサイクルの速度が向上してきています。それについて説明しましょう。例として，

図5に4クロック(68000)，3クロック(68020/68030)，2クロック(68030)，バースト(68030)のリードサイクルのタイミング図を示します。

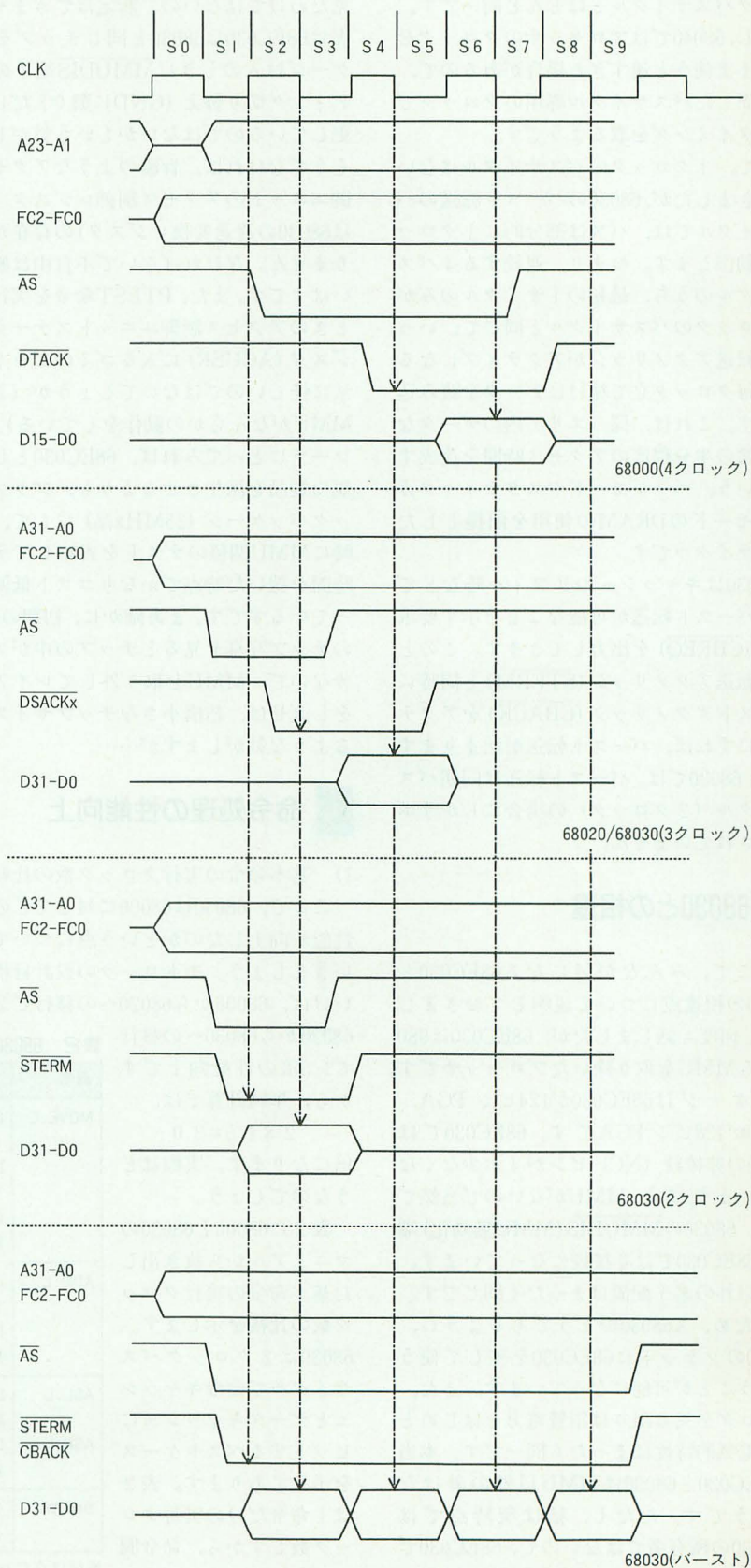
ところで，68000ファミリのバスサイクルには非同期，同期という言葉が出てきます。非同期とはバスのタイミングがクロックに同期していないデータストローブなどの信号に従って行われるという意味です。逆に同期とはバスタイミングがクロックに同期しているという意味です。同期，非同期という言葉を深く考える必要はありません。どちらも，バスサイクル期間中に転送アクノリッジ（レディ信号）をサンプリングして，それがアクティブならばバスサイクルを終結させ，インアクティブならウェイトを挿入するといった，ごく普通のバスサイクルです。バスサイクルが短くなるにつれて転送アクノリッジのサンプリングは早くなっているだけです。

最初の68000では非同期の4クロックバスサイクルが採用されています。アドレスバスはS0期間でいったんハイインピーダンスにしてから駆動します。その後，S4の終わりで転送アクノリッジ($\overline{\mathrm{DTACK}}$)をサンプリングし，アクティブなら1クロック後のS6の終わりでデータを取り込みます。いま見るとハエの止まりそうなバスサイクルです。

それじゃあ，あんまりだということで68020ではバスサイクルが3クロックになりました。68000に比べてアドレスバスの駆動を半クロック前倒しし，転送アクノリッジ（$\overline{\mathrm{DSACKx}}$）のサンプリングを1クロック手前に持ってきました。これで1クロックの節約です。

続く68030では，68020との互換性維持のために3クロックのバスサイクルを残しつつ，2クロックのバスサイクルに挑戦しています（ただし同期バスで接続した場合のみ)。これは，クロックの場合よりもさらに半クロック転送アクノリッジ($\overline{\mathrm{STERM}}$)のサンプリングを前倒ししています。そして，転送アクノリッジがアクティブになってからのデータ取り込みを1クロック後から半クロック後に変更しました。これで1クロックの節約です。

この2クロックサイクルの次には1クロックサイクルというのも考えられます。しかし，これはクロックの周期とつながるメモリのアクセス時間を考えればほとんど無茶な世界になってきます。そこで，68000ファミリではいまのところ(68040までは)バスサイクルは2クロックということで落ち

図5　各プロセッサのリードサイクル

着いているようです。図には書いていませんが，68040のバスサイクルは68030の２クロックバスサイクルとほとんど同一です。ただし，68040ではプロセッサのクロックをそのまま使うと速すぎる場合があるので，２分周したバスサイクル専用のクロックでバスタイミングを取るようです。

さて，１クロックのバスサイクルはないと書きましたが，68030のバースト転送のバスサイクルでは，バスは部分的に１クロックで動作します。つまり，連続する４バスサイクルのうち，最初の１サイクルのみが２クロックのバスサイクルと同じで，いったん転送アクノリッジがアクティブになると，毎クロック立て続けにデータを読み込みます。これは，同一エリア内のデータなら通常の半分程度のアクセス時間を達成するという，ページモードやスタティックカラムモードのDRAMの使用を前提としたバスサイクルです。

68030はキャッシュのリフィル時などでは，バースト転送が可能なことを示す要求信号（$\overline{\text{CBREQ}}$）を出力してきます。このとき，転送アクノリッジ（$\overline{\text{STERM}}$）と同時にバーストアクノリッジ（$\overline{\text{CBACK}}$）をアクティブにすれば，バースト転送が始まります。なお，68030では，バースト転送は同期バスサイクル（２クロック）の場合にしかサポートされていません。

68030との相違

ここで，みんなが気になる68EC030と68030の相違点について説明しておきましょう。何度も話しましたが，68EC030は68030からMMUを取り除いたプロセッサです。パッケージは68EC030が124ピンPGA，68030が128ピンPGAです。68EC030では68030の非接続（NC）ピンが４本少なくなっています。あと，MMUがないので当然ですが，68030の$\overline{\text{MMUDIS}}$（MMU無効化）端子が68EC030では非接続になっています。それ以外の端子配置はまったく同じです。このため，X68030がそうであるように，68030のソケットに68EC030を差して使うということが可能になっています。また，カタログを見る限りは消費電力をはじめとする電気的特性はまったく同一です。本当に68EC030と68030はMMU以外の差はないようです。ただし，私は現時点ではX68030の所有者ではないので，68EC030では未定義になった68030のMMU制御命令を実行すると何が起きるのかはわかりません（多分無視されるだけでしょう）。

ところで，68EC030には本当にMMUがないのでしょうか。パッケージを開封して見たわけではないので断定はできませんが，実は68EC030は68030と同じチップをパッケージ封入のときに$\overline{\text{MMUDIS}}$端子のボンディング切り替え（GNDに繋ぐ）だけで変更しているのではないかという気がします。そうでなければ，盲腸のようなアクセス制御ユニットのアクセス制御レジスタ（これは68030の透過変換レジスタ）の存在がわかりません。なければないで不自由は感じないはずです。また，PTEST命令を実行したときのアクセス制御ユニットステータスレジスタ（ACUSR）に入るゴミのビットも非常に怪しいのではないでしょうか（きっとMMUがなんらかの動作をしている）。モトローラにとってみれば，68EC030として新規に製品を派生させるよりも，プラスティックパッケージ（25MHz品）にして，出荷時にMMU関係のテストを省略してテスト時間を稼いだ時点でかなりコスト低減になっているずです。まあ確かに，初期の68030のチップ写真を見るとチップの中がスカスカなので，MMUを取り外してレイアウトをし直せば，結構小さなチップサイズになるような気がしますが……。

命令処理の性能向上

1）　基本命令の実行クロック数の比較

ここで，68030は68000に対してどの程度性能が向上したのかという点について見ていきましょう。モトローラの設計目標からいけば，68000から68020への移行で２倍，68020から68030への移行で1.5倍の性能向上ですから，単純計算では，

$$2\times1.5=3.0$$

倍になります。実際はどうなのでしょう。

表２に68000と68030のマニュアルから抜き出した基本命令の実行クロック数の比較を示します。68030は２クロックバスサイクルで命令キャッシュとデータキャッシュにヒットするベストケースを示してあります。表２は１命令だけの実行クロック数ですから，命令間のオーバーラップによる実行クロックの減少に関しては考慮してありません。実効アドレスフェッチによる命令内でのオーバーラップは考慮してあります。

表2　68030と68000の性能比較

命令	アドレッシング	68030	68000	性能比１	性能比２
MOVE.L	Dn, Dm	2	4	3.1	2.1
	(An), Dm	5	12	3.8	2.5
	Dn, (Am)	3	12	6.3	4.2
	(An), (Am)	6	20	5.2	3.5
	#imm, Dn	6	12	3.1	2.1
	#imm, (Am)	8	20	3.9	2.6
ADD.L	Dn, Dm	2	8	6.3	4.2
	(An), Dm	5	14	4.4	2.9
	Dn, (Am)	6	20	5.2	3.5
	#imm, Dn	6	16	4.2	2.8
ASL.L	Dx, Dy	8	8+2n	1.6-14.5	1.0- 9.6
	#imm, Dx	6	8+2n	2.1-19.3	1.4-12.8
ASR.L	Dx, Dy	6	8+2n	2.1-19.3	1.4-12.8
	#imm, Dx	4	8+2n	3.1-28.9	2.1-33.0
Bcc.W	（分岐）	6	10	2.6	1.7
	（不分岐）	6	12	3.1	2.1

単位はクロック数。
68030は命令キャッシュ，データキャッシュにすべてヒットすると仮定。
性能比１は68030（25MHz）が68000（16MHz）の何倍高速かを計算。
性能比２は68030（25MHz）が68000（24MHz）の何倍高速かを計算。

表２によると25MHzの68030は16MHzの68000に対して３～６倍は速いはずです。マニアによって改造された24MHzの68000に対しては２～４倍でしょう。ただし，これは本当にベストケースであることを忘れてはいけません。実際のアプリケーションを動かした経験では，たかが256バイトの，しかもダイレクトマップ方式のキャッシュによる性能向上は大きくないように思えました。３～６倍という値はもっと割り引いて考えなければならないでしょう。

シャープの公称値によると，Dhrystoneベンチマークによる性能値がキャッシュをオンにしたときに16MHzの68000（XVI）に比べて2.4倍ということですから，たとえ生のチップとパソコンというシステムの差はあれ，そんなものかなというのが素直な感想です。

2）　命令実行のオーバーラップ

ところで，68030は命令パイプというものを持っていますから，（キャッシュにヒットし続ける限りは）命令の実行をある程度オーバーラップして実行することができます。基本的な考え方としては，シーケンサがオペランドを読み込む準備（アドレス計算）をしていてバスコントローラが暇にしている時間と，バスコントローラがライト動作をしているためにシーケンサが暇にしている時間がオーバーラップできる時間の対象になります。

68030のマニュアルでは，先のバスコントローラが暇にしている時間をヘッド（頭），

シーケンサが暇にしている時間をテイル(尻尾)といっているようです。68030ではアドレス計算と命令実行を別々にマイクロプログラムで行うようなので，アドレス計算と命令実行のそれぞれにヘッドとテイルがあるようです。

命令の実行を飛び越すことはできないという性格上，

1命令前の命令実行のテイル
↓↑
実効アドレスフェッチのヘッド

実効アドレスフェッチのテイル
↓↑
命令実行のヘッド

命令実行のテイル
↓↑
次の命令の実効アドレスフェッチのヘッド

という組み合わせがオーバーラップできる対象になります。このとき，オーバーラップできるクロック数は，たかだかオーバーラップ可能な組のヘッドとテイルのクロック数の最小値ということになります。片方でたくさんオーバーラップできるようになっていても，他方にそれだけの余裕がなければなりませんから当たり前ですね。

実例でオーバラップを計算してみましょう。次のような命令列を考えます。

```
move.l  d0,-(sp)
add.l   d1,(a1)
move.l  d0,d1
move.l  (a2),(a3)
```

まず，これらの命令の(キャッシュケースの)実行クロック数，ヘッド，テイルのクロック数を調べます。アドレス計算を伴うような命令では，実効アドレスのフェッチ時間，あるいは実効アドレスの計算時間を加えるように指示されているので，それも併せて抜き出します。

これをまとめたのが表3です。時間的に隣り合うヘッドとテイルの最小値を考慮しながら，命令実行の様子を図示したものが図6になります。図6を見れば明らかなように，本来ならば18クロックかかる命令実行が16クロックで終了しています。うーん，思ったほどはオーバーラップしていませんね。

3) 実行時間0の命令

68020や68030のマニュアルを見ると，オーバーラップがうまくいくと，そこに命令の実行時間が吸収されて実質的に実行時間が0になるという記述があります。そういった理由で，68020のマニュアルでは，ヘッドの時間が実行時間に等しい命令についてはは実行時間のベストケースが0になっていました。しかし，その表現は68030のマニュアルでは訂正されています。68030のマニュアルの実行クロック数の表を見ると，そういう場合はほとんどありえないことがわかるからです。

そこでは，(実効アドレスフェッチを除く)ほとんどの命令のテイルのクロック数は0か1となっています。これとオーバーラップさせ，実質的に時間0で命令を実行させるためには，ヘッドが1の命令が必要です。一般的にはヘッドのクロック数は命令実行のクロック数より大きくなれません(命令フェッチと同時に計算が行われるイミディエートアドレッシングなどに例外がありますが)。すなわち，実効アドレス計算以外を1クロックで実行する命令が必要です。ところが68030の命令実行の最小クロック数は2ということになっていますから，このようなことは普通は不可能です。

おわりに

68030の概要についてざっと見てきました。ソフトウェアに関する概要はいろんな人が知っているし，ほかの記事で書いているはずなので，今回はちょっと違った観点から解説してみました。まとまりに欠ける部分があると思いますが，それは試行錯誤の結果ということで勘弁してください。

それはそうと，パソコンにとって本当にMMUは不要なのでしょうか。最近のMacintoshの傾向を見ていると，ほとんどが68040を採用していますし，廉価版を採用する場合でもMMUだけは使用できる68LC040になっています。

これらのシステムでMMUが実際に活用されているのかは疑問な点もありますが，MMUというのは趣味で遊ぶのには興味深いテーマです。パソコンだからMMUが不要というのはあまりにも常識的すぎる論理です(ひょっとしたらOSが間に合わなかったという理由かもしれませんが)。パソコンだからこそ，現実との妥協で変な使用上の制限をつけずに，自由度の高い環境を提供してくれてもよかったのではと思います。

P.S. もし，X68030にMMUがあれば「要2Mバイト」というアプリケーションソフトは姿を消してしまったことでしょう(あっ，X68030は4Mバイト内蔵だった。じゃあ次は「要12Mバイト」かな)。

《参考文献》

1) モトローラ，ユーザーズ・マニュアル MC68030，1990年.
2) モトローラ， M68000マイクロプロセッサ ユーザーズ・マニュアル 第4版，1984年.
3) モトローラ， MC68020 ユーザーズ・マニュアル 第2版，1986年.
4) MOTOROLA，M68000 FAMILY REFERENCE，1990.
5) MOTOROLA，PROGRAMMER's REFERENCE MANUAL，1992.

表3 命令実行クロックの例

命令	内訳	ヘッド	テイル	クロック
move.l d0,-(sp)	op	0	2	4
add.l d1,(a1)	fea	1	1	3
	op	0	1	3
move.l d0,d1	op	2	0	2
move.l (a2),(a3)	fea	0	1	3
	op	2	0	4

fea …実効アドレスフェッチ時間
op …命令実行時間

図6 命令実行のオーバーラップ

命令列	→ 時間	実行クロック	実効的クロック
move.l d0,-(sp)	■■■■	4 →	3
add.l d1,(a1)	□□□ ■■■	6 →	5
move.l d0,d1	■■	2 →	2
move.l (a2),(a3)	□□□ ■■■■	6 →	6
合計クロック数		18 →	16

□ 実効アドレスフェッチ
■ 命令実行

プログラミングの予備知識

MC68030の使い方

X68030

Murata Toshiyuki 村田 敏幸

ひと昔前ならワークステーションクラスの32ビットマシンでアセンブラを駆使するという贅沢。68030を使った国産パソコンはこれまで存在しませんでした。そこには前人未踏の世界が広がっています。

68000ファミリプロセッサの系譜

モトローラ68000は外部データバスは16ビットながら，内部は32ビットプロセッサそのもので，16本の32ビット汎用レジスタ，直交性が高く，高水準言語との親和性に優れた命令セット，リニアでフラットな16Mバイトのアドレス空間，ユーザーモード/スーパーバイザモードという特権レベル区分によるメモリ保護などを特徴とする，いま見てもよくできたマイクロプロセッサだ。それでも，強いて挙げるなら，当初からワークステーションクラスのコンピュータに使われることを意識していたにもかかわらず，本格的なマルチタスクには不可欠な仮想記憶を容易には実現できないという欠点があった。

その後，この点を改良した68010，本格的な32ビットCPUとなった68020，その高速版といえる68030，さらに3～5倍性能を上げた68040という怪物CPUが発表され現在に至っている。

で，特筆すべきは，68000から68040まで，ソフトウェア面のアーキテクチャはまったく変更されておらず，オブジェクトコードレベルでの互換性が保たれている点だ。命令セットは68020の時点で拡充され，以後も若干の増減はあるものの，ほとんどの部分は68000から68040までそのまま受け継がれている。68040ぐらいになると中身は68000とは似ても似つかないというのに，68000用に書いたプログラムは68040でも基本的にはそのまま動く。初期設計の貧弱さを隠すため，いたずらに特定の用途/場面でのみ使える専用命令を追加したり，新たなプログラミングモデルを導入したりしたうえで，時代遅れの互換モードを残してお茶を濁すのとはわけが違うのだ。

さて，X68030には68030の姉妹品である68EC030が採用されている。MMUが外されている以外は，命令セットはもちろん，命令キャッシュ/データキャッシュ，命令実行のオーバーレイなど，機能/性能とも68030そのものだ。ふつうならワークステーションクラスに使われ，OSのベール越しにしか触れることのできなかった68030を好き勝手に使える幸福をかみしめつつ，以下，68030のソフトウェア面を概観してみたい。

レジスタセット

図1に68EC030のレジスタセットを示す。68000でお馴染みのものも多いが，ひととおり紹介しておこう。

●データレジスタ（D0～D7）

主としてデータ操作/演算に用いられる，8本の32ビット汎用レジスタ。8本が完全に対等で，すべてがアキュムレータとなりうる。32ビットをまとめて扱う以外に，下位8ビットまたは16ビットだけを切り出して使うことができる。以上の用途に加え，68020以降，アドレッシングモードが拡張されたことにより，メモリの間接指定にも使用できるようになった。

●アドレスレジスタ（A0～A7）

主としてメモリアドレスの間接指定に用いる8本の32ビットレジスタ。うち，A7はシステムスタックポインタとして用いられる。スタック操作は専用命令ではなく，単にアドレスレジスタA7に対して適切なアドレッシングモードを適用することで実現される。逆にいうと，スタックポインタもほかのアドレスレジスタ同様に多くの命令で使用できる。

68000ではメモリ空間が16Mバイトまでだったため，アドレスレジスタの上位8ビットは意味を持たなかったが，68020以降メモリ空間が4Gバイトに拡張されたのに伴い，全32ビットが有効となっている。ただし，X68030ではアドレスの上位8ビットはデコードされていないようで，事実上，これまでどおり下位24ビットのみが意味を持つ。Human68kやSX-WINDOWやXCのライブラリでは，ときにアドレスの最上位バイトを別の目的で使っている場合があるため，互換性確保のためにやむをえなかったのだろう。ただ，論理アドレスのレベルでは32ビットすべてが意味を持つはずだから，将来予想される68EC030→68030の移行を考慮すると，アドレスレジスタの上位8ビットにゴミを入れたままメモリアクセスするのは避けたほうがよいと考えられる。

●スタックポインタ（A7＝SP）

68000はユーザースタックポインタ（USP）とスーパーバイザスタックポインタ（SSP）の2本のスタックポインタを持ち，現在の特権モード（SRのSビット）に応じて，両者が切り替わる。アドレスレジスタA7は実体を持たない窓口であり，ユーザーモードではUSPに，スーパーバイザモードではSSPに通じている。68020以降では，このSSPがさらにISPとMSPに分かれた。

ISP（割り込みスタックポインタ）はその名のとおり割り込みハンドラ用のスタックポインタであり，ハードウェア割り込みがかかったときに使われる。MSP（マスタスタックポインタ）は，通常のスーパーバイザタスクが使う。割り込み発生時に自動的にスタックポインタがISPに切り替わるので，ふつうのタスクは割り込みハンドラの使用する分を考慮してスタックを大きくとる必要がない。ただ，Human68kでは，常にISPのみが使われ，MSPは利用していないようだ。割り込み時のMSP⟷ISPの切り替えは少々余分な操作を伴うので，シングルタスクのHuman68kでは，これはこれでよいように思う。

●プログラムカウンタ（PC）

次に実行する命令のアドレスを保持する32ビットレジスタ。アドレスレジスタ同様，68020以降では全32ビットが有効だが，

X68030では事実上，下位24ビットのみが意味を持つ。

●ステータスレジスタ（SR）/コンディションコードレジスタ（CCR）

SRはプロセッサの各種ステータスを保持する16ビットレジスタ。図2のようなビットフィールドになっており，ビットごとに意味を持つ。CCRはSRのユーザーバイトにつけられた別名で，直前の演算結果を保持する。ユーザーモードではCCRにのみアクセスできる。

図2にも示したように，68020以降のCPUではSRの機能が一部拡張されている。まず，デバッガでプログラムのシングルステップ実行を実現するためのトレースイネーブルだが，68000ではシングルステップ実行をするかしないかをTビットの1ビットで表していたのに対して，68020以降“分岐命令のみトレース”という技が使えるようになり，T0，T1の2ビット構成になった（表1）。

また，Mビットは前述のISPとMSPの切り替えスイッチで，1のときMSPが使用される。

●ベクタベースレジスタ（VBR）

VBRは68010で仮想記憶対応の一環として新設された32ビットレジスタだ。68000では例外ベクタテーブルは必ず0番地から始まることになっていたが，68010以降，必要に応じて複数の例外ベクタテーブルを用意することができるようになった。VBRはその先頭アドレスを保持する。もっとも，X68030（というよりHuman68k）ではいまのところ例外ベクタテーブルは0番地固定で，VBRは常に0になっている。

●オルタネートファンクションコードレジスタ（SFC，DFC）

SFCレジスタとDFCレジスタもまた68010で新設された，各3ビットのレジスタだ。このレジスタはMOVES命令でのみ使用される。MOVESはアドレス空間をまたいでデータ転送を行う特権命令（スーパーバイザモードでのみ使用可能な命令）で，SFCはその転送元空間，DFCは転送先空間を指定するためにある（表2）。

X68030では，スーパーバイザ/ユーザー各モードのプログラム空間＝データ空間で，かつ，スーパーバイザ空間がユーザー空間を包含しているため，このレジスタの存在価値は見えてこないだろう。実際，使い道もなさそうだ。が，キャッシュとの絡みで，ファンクションコードというものの存在は，ちょっと頭に入れておいてもらいたい。

●キャッシュコントロールレジスタ（CACR）/キャッシュアドレスレジスタ（CAAR）

CACRとCAARは68020で追加された32ビットレジスタで，内蔵キャッシュの制御をする。68030ではデータキャッシュが内蔵

図1　68EC030のレジスタ

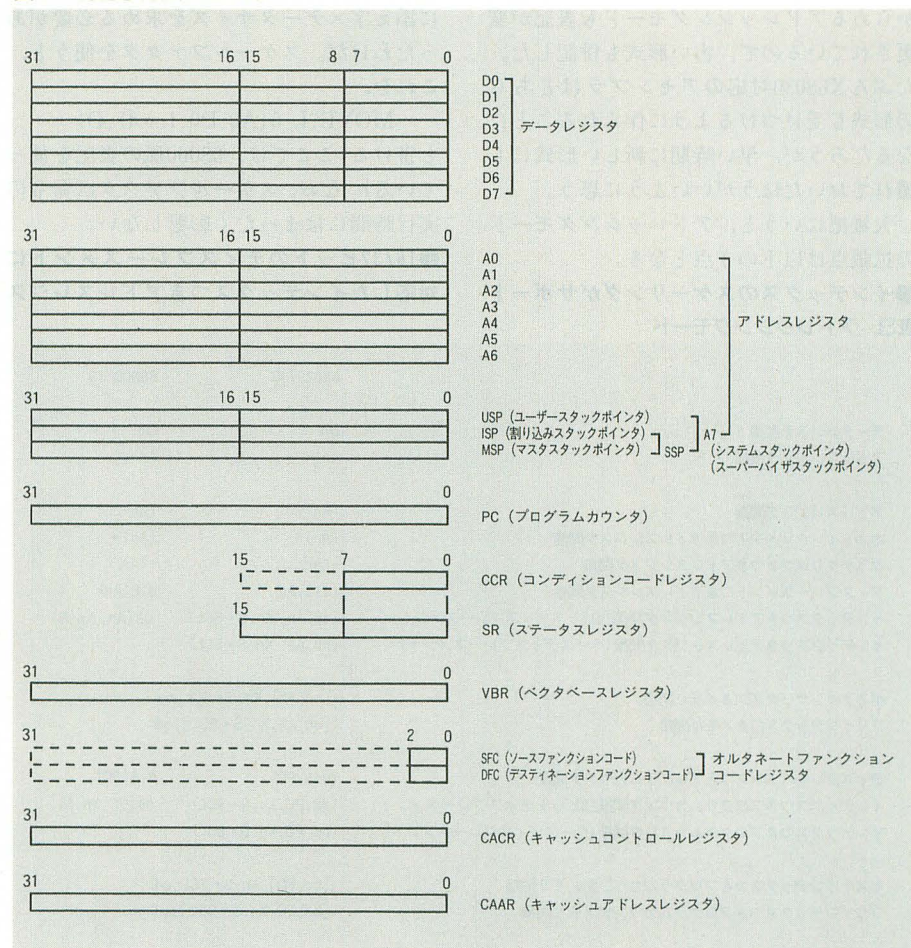
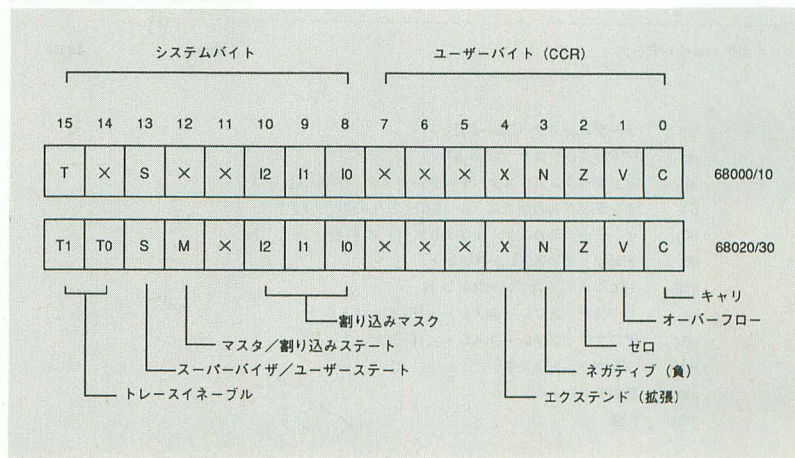



表1　トレースイネーブルビット

T1	T0	
0	0	トレースなし
0	1	分岐命令のみトレース
1	0	全命令トレース（68000互換）
1	1	無効

表2　ファンクションコード

FC2	FC1	FC0	
0	0	0	（予約）
0	0	1	ユーザーデータ空間
0	1	0	ユーザープログラム空間
0	1	1	（空き）
1	0	0	（予約）
1	0	1	スーパーバイザデータ空間
1	1	0	スーパーバイザプログラム空間
1	1	1	CPU空間（コプロセッサなどとの通信に利用）

図2　ステータスレジスタ

されたことに伴い，CACRの機能が増えている。詳しくは，キャッシュの項で触れよう。

アドレッシングモード

プログラムを書く立場で68000と68030を比べたとき，命令セットの追加よりもおいしいのが，アドレッシングモードの拡張だ。表3に68020以降で有効なアドレッシングモードの一覧を示す。わけあって68000時代からあるアドレッシングモードも表記が変更されているので，古い形式も併記した。たぶんX68030対応のアセンブラはどちらの形式も受けつけるように作られることになるだろうが，早い時期に新しい形式にも慣れておいたほうがいいように思う。

大雑把にいうと，アドレッシングモードの拡張点は以下の3点となる。

●インデックスのスケーリングがサポートされた

インデックスに対して，1，2，4，8のいずれかのスケールファクタを指定すると，アドレス算出の際にその値が乗じられる。スケールファクタは配列参照のときに重宝する。たとえば，68000では4バイトデータの配列に対するアクセスを，

```
LSL.L    #2,D0
MOVE.L 0(A0,D0.L),D1
```

で実現した（A0に配列先頭アドレス，D0.Lに添え字が入っているものとする）。事前に添え字×データサイズを求める必要があったわけだ。スケールファクタを使うと，これを，

```
MOVE.L 0(A0,D0.L*4),D1
```

と書ける（ここでは，68000風の表記を使っている）。なお，スケールファクタは命令の実行時間にはまったく影響しない。

●16/32ビットのディスプレースメントに対応したインデックスつきアドレスレジスタ/プログラムカウンタ間接形式がサポートされた

68000のインデックスつきレジスタ間接形式では，ディスプレースメントが8ビットに制限されていたため，ときに窮屈な思いをしたが，68020以降，16/32ビットのディスプレースメントが使えるようになった。

●メモリ間接形式がサポートされた

これはいわゆる2重間接（ポインタのポインタ）を実現するもので，指定のアドレスに格納された32ビット値をアドレスとみなして，もう1回間接参照するアドレッシングモードだ。ベースレジスタとしてアドレスレジスタを使うかプログラムカウンタを使うかの2系統，プリインデックス/ポストインデックスの2種類，都合4種類が用意されている。

プリインデックスとポストインデックスの違いは，アセンブリ言語のシンタックスを見比べれば一目瞭然だろう。プリインデックスのメモリ間接形式では，インデックスは2重間接の1段目のアドレス算出に使われる。ポストインデックスのほうは2段目で使われる。つまり，“[]”の中が1段目のアドレスとなり，そのアドレスから取り出した32ビット値に“[]”の外の要素を加えたアドレスが最終的な実効アドレスとなる。

＊

さて，新設された“長いディスプレースメント”対応のインデックスつきレジスタ間接形式は見かけ以上に強力だ。この新しいアドレッシングモードはジェネリックな(generic：汎用の，総称的な）レジスタ間接形式として実装されている。この形式では，

・ディスプレースメント（16/32ビット）
・ベースレジスタ（アドレスレジスタかプログラムカウンタ）
・インデックスレジスタ（データレジスタかアドレスレジスタ）×スケールファクタ

の3つの要素でアドレスを指定するわけだが，そのいずれもがサプレス可能なのだ。不要な要素は省略できる（同様に，メモリ間接形式も，4つの値すべてが省略可能）。これにより，さまざまなバリエーションが利用できることになる。

たとえば，68020以降でも（インデックスのつかない）ディスプレースメントつきアドレスレジスタ間接形式のディスプレースメントは16ビットに制限されているが，(bd,An,Xn.S*SCL)のインデックスを省略すれば，

(bd,An)

表3　アドレッシングモード

	68020/30	68000/10
データレジスタ直接	Dn	Dn
アドレスレジスタ直接	An	An
アドレスレジスタ間接	(An)	(An)
ポストインクリメントつきアドレスレジスタ間接	(An)+	(An)+
プリデクリメントつきアドレスレジスタ間接	-(An)	-(An)
ディスプレースメントつきアドレスレジスタ間接	(d16,An)	d16(An)
インデックスつきアドレスレジスタ間接(8ビットディスプレースメント)	(d8,An,Xn.S*SCL)	d8(An,Xn.S)
インデックスつきアドレスレジスタ間接(ベースディスプレースメント)	(bd,An,Xn.S*SCL)	
ポストインデックスつきメモリ間接	([bd,An],Xn.S*SCL,od)	
プリインデックスつきメモリ間接	([bd,An,Xn.S*SCL],od)	
ディスプレースメントつきプログラムカウンタ間接	(d16,PC)	d16(PC)
インデックスつきプログラムカウンタ間接(8ビットディスプレースメント)	(d8,PC,Xn.S*SCL)	d8(PC,Xn.S)
インデックスつきプログラムカウンタ間接(ベースディスプレースメント)	(bd,PC,Xn.S*SCL)	
ポストインデックスつきプログラムカウンタメモリ間接	([bd,PC],Xn.S*SCL,od)	
プリインデックスつきプログラムカウンタメモリ間接	([bd,PC,Xn.S*SCL],od)	
絶対ショートアドレス	(abs).W	abs.W
絶対ロングアドレス	(abs).L	abs.L
イミディエイトデータ	#data	#data

Dn	データレジスタ（D0～D7）
An	アドレスレジスタ（A0～A7）
Xn	インデックスレジスタ（データレジスタ/アドレスレジスタ）
S	インデックスのサイズ（W=16ビット/L=32ビット）
SCL	インデックスのスケールファクタ（1,2,4,8）
d8	8ビットディスプレースメント
d16	16ビットディスプレースメント
bd	ベースディスプレースメント（16/32ビット）
od	アウタディスプレースメント（16/32ビット）
PC	プログラムカウンタ
abs	絶対アドレス
data	即値

となり，(d16, An)が32ビットディスプレースメント対応になったのと同じ感覚で使える。また，インデックス以外をすべて省略して，インデックスにデータレジスタを使った，

(Dn.L)

は，事実上，“データレジスタ間接”になるし，同じノリで“ディスプレースメントつきデータレジスタ間接”も実現できる。データレジスタもポインタとして使えるわけだ（ただ，素直にアドレスレジスタを利用する場合よりは遅い）。

ここで，“.L”がついていることに注意したい。インデックスは16ビットの場合と32ビットの場合があり，サイズを明示しないとアセンブラは16ビットとみなす。これを忘れて，

(Dn)

と書いてしまうと，

(Dn.W)

と解釈され，Dn.Lの指すアドレスではなく，Dn.Wを32ビットに符号拡張したアドレスを参照するコードが生成される。

ところで，

(bd, An, Xn.S*SCL)
(bd, PC, Xn.S*SCL)

のベースレジスタを省略すると，ともに，

(bd, Xn.S*SCL)

となるが，68000ファミリプロセッサではアドレスレジスタ間接はデータ空間，プログラムカウンタ間接はプログラム空間の参照となり，両者は区別される。アセンブリ言語レベルでも区別する方法がなければならない。そこで，アドレスレジスタを省略する場合は，そのまま，

(bd, Xn.S*SCL)

と書き，PCを省略する場合にはZPCというシンボルを使って，

(bd, ZPC, Xn.S*SCL)

と書くことで，“プログラム空間の参照だが，PCの代わりに値0を使う”ことをアセンブラに指示する。もっとも，X68030ではプログラム空間＝データ空間だから両者の区別に意味はなく，ZPCを明示してプログラム空間を絶対アドレスで参照する（PCをサプレスするというのはそういうこと）機会はないだろう。

命令セット

命令セットも68020の段階でかなり拡張された。便利そうな命令が新設されたのはもちろん，既存命令もきちんと32ビット化されている。

●BRA, Bcc, BSR

これまでの8/16ビットディスプレースメントに加え，32ビットディスプレースメントがサポートされた。これにより，全メモリ空間の任意のアドレスにプログラムカウンタ相対で分岐できることとなり，リロケータブルなプログラムがより作りやすくなった。

なお，68000ではディスプレースメントが8ビットか16ビットかを命令ワード中の8ビットディスプレースメント格納フィールドが00_Hかどうかで区別していた（00_Hなら後続の1ワードが真の16ビットディスプレースメント）が，68020以降，ここがFF_Hなら32ビットディスプレースメントと解釈される。

●MULS, MULU, DIVS, DIVU, DIVSL, DIVUL

32ビットの乗除算がサポートされた。乗算命令MULS/MULUでは，サイズにロングワードを指定した，

MULS.L <ea>, Dn
MULU.L <ea>, Dn

により，Dn.Lに32ビット×32ビットの積が得られる。32ビット×32ビットの積は最大64ビットになりうるが，この形式では上位32ビットは捨てられる。結果を64ビットで受けたい場合は，

MULS.L <ea>, Dh:Dl
MULU.L <ea>, Dh:Dl

のように，“:”を挟んで並べたデータレジスタのペアを指定する。この場合，Dl.Lが被乗数となり，積の上位32ビットがDh.Lに，下位32ビットがDl.Lに得られる。

除算命令DIVS/DIVUでは，サイズにロングワードを指定した，

DIVS.L <ea>, Dn
DIVU.L <ea>, Dn

により，Dn.Lに32ビット÷32ビットの商を得ることができる。この形式では余りは捨てられる。余りも得たければ，新設されたDIVSL/DIVUL命令を使い，

DIVSL.L <ea>, Dr:Dq
DIVUL.L <ea>, Dr:Dq

のようにすると，Dq.Lを割った結果の商がDq.Lに，余りがDr.Lに求まる。さらに，

DIVS.L <ea>, Dr:Dq
DIVU.L <ea>, Dr:Dq

により，Drを上位ロングワード，Dqを下位ロングワードとする64ビット÷32ビットの商と余りがDq.LとDr.Lに得られる。レジスタペアを使ったDIVS.L/DIVU.LとDIVSL.L/DIVUL.Lは混用しないよう，いまから注意しておきたい。

●CMPI, TST

68000系プロセッサでは，プログラムは実行時に書き換わるべきではないという設計理念から，PC相対のアドレッシングモードをデスティネーションオペランドに適用することが許されていない。68000ではこのルールが厳格に適用されており，読み込むだけで書き換えるわけではないCMPI，TSTの比較オペランドでもPC相対形式が使えなかった。68020以降では，これが許される。

●シフト/ローテート命令

機能は変わっていない。バリエーションも増えていない。ただ，68000ではシフト/ローテートするビット数が増えるにしたがって実行時間が長くなったのに対して，68020以降ではバレルシフタの採用により，何ビットシフトしようと1ビットの場合と同じ時間しかかからない。ビットフィールド命令が新設されたこともあり，ビット列の操作は楽になりそうだ。

●LINK

32ビットディスプレースメントがサポートされた。アドレッシングモードの拡張とあいまって，スタック上に大きなローカル変数領域をとるのが容易になった。

●RTD

RTD（ReTurn and Deallocate）はいわゆるPascal風の関数呼び出しを実現する命令だ。

RTD #disp

のように使い，スタックトップからリターンアドレスを取り出すと同時に，スタックポインタに指定の値を加える。スタックを介してサブルーチンに引数を渡す場合のスタックの掃除をまとめてやってくれるわけで，プログラムの速度とサイズの両面で貢献するだろう。

●MOVE from CCR

68010で追加された，CCRからのデータ転送を行う命令だ。サブルーチンを，

```
MOVE.W   CCR,-(SP)
MOVEM.L  D0-D7/A0-A6,-(SP)
    :
MOVEM.L  (SP)+,D0-D7/A0-A6
RTR
```

のように構成することで，CCRを含む全レジスタが保存できる。68000では，同様の場面でCCRをセーブするのに，

MOVE.W SR,-(SP)

を使った。CCRはSRの下位バイトに与えられた別名であり，RTRはスタックトップから1ワードを取り出すものの上位バイトは捨てて，下位バイトをCCRに転送するの

で，これでよかった。しかし，SRがユーザーモードでも参照できてしまうのは邪道ということで，68010以降，こちらは特権命令となっている。

こうなると困るのは68000用に書かれたMOVE from SRを使ったプログラムだ。68010以降を使ったマシン上で動かすと特権違反にされてしまう。が，多くのシステムでは，特権違反例外ハンドラが気をきかすことでこの問題を解決している。つまり，ユーザーモードでMOVE from SRが使われたら，CCRの値を得たかったのだろうと解釈して，特権違反を起こした該当命令をMOVE from CCRに書き換え，再実行するのだ。もちろん，Human68kもそうなっている。

●BFxxxx

68020での新設命令のうち，数的に大きな部分を占めるのが，最大5バイトにまたがる32ビットまでのビットフィールドを扱う命令群だ。C言語のビットフィールドが落としやすくなるのはもちろん，水平型VRAM(X68030ならテキスト画面）の操作にも便利だろう。

ビットフィールド命令としては，次のものが用意されている。

BFCHG	ビット反転
BFCLR	クリア
BFEXTS	符号つき抽出
BFEXTU	符号なし抽出
BFFFO	セットされているビットの検索
BFINS	挿入
BFSET	1で埋める
BFTST	0との比較

ビットフィールドは，

<ea> {offset : width}

の形で，アドレス<ea>のoffset(負の値も有効）ビット目からwidthビットという具合に指定する。ここで，ビットフィールド命令におけるビット番号は，通常の場合と逆になっていることに注意したい。

●PACK，UNPK

PACKは，バイアスのかかった1桁/バイトのアンパック形式BCDを2桁/バイトのパック形式BCDに変換し，UNPKはその逆変換を行う。わざと変な表現を使ってみたのだが，バイアスのかかった1桁/バイトのBCDとは，早い話が数値文字列だ。文字コード体系によらず，数字には連続した文字コードが割り当てられているもので，ASCIIの場合“0”～“9”が30_H～39_Hに対応している。したがって，PACK/UNPKにおいて，バイアス値として“0”の文字コードである30_Hを指定することで，2桁/バイトのBCDと数値ASCII文字列の相互変換が行える。

●CMP2

CMP2はレジスタの値が特定の範囲に収まっているかどうかを調べる命令だ。テスト結果はCCRのCビットに反映され，範囲内のときC＝0，範囲外のときC＝1となる。また，ちょうど境界上だった場合はZビットが立つ。アセンブリ言語での書式は，

CMP2 <ea>，Rn

のようになり，<ea>で指定したアドレスから下限値，上限値を並べておく。比較を符号つきで行うか，無符号で行うかを指定するかは，下限値と上限値の関係で決まるらしい。たとえば，サイズがワードのとき，

下限値＝$FFFF_H$
上限値＝0001_H

なら，－1～1として扱われ，比較は符号つきで行われる。これを逆にして，

下限値＝0001_H
上限値＝$FFFF_H$

とすると，1～65535とみなされ，無符号比較が行われる。

●EXTB

EXT命令のバリエーションで，Dn.Bの8ビットを32ビットに符号拡張してDn.Lに収める。これまでは，

```
EXT.W  D0
EXT.L  D0
```

の2命令で行っていた操作が，

```
EXTB.L  D0
```

一発で行える。

●コプロセッサ命令

コプロセッサとの通信プロトコルを自動的に行う命令だ。X68030の場合，オプションの68882を装着することで，コプロッサ命令を使った浮動小数点演算が行えるようになる。もっとも，当面は既存の機種との互換性を保つため，68882を直接ドライブするのはそれがどうしても必要なプログラム以外では避けて，なるべくFLOATn.Xのファンクションコールを使用するべきだろう。

なお，コプロセッサ命令周りでは，Human68kのDOSコールとの絡みで少々やっかいな事態が発生したようだ。詳しくはコラムのほうを参照してもらいたい。

●MOVES

アドレス空間をまたいだデータ転送を行う特権命令。すでに触れたように，X68030では使う意味がないだろう。

●MOVEC

特殊制御レジスタと汎用レジスタ間のデータ転送を行う特権命令。特殊制御レジスタとしては，VBR，SFC，DFC，CACR，CAAR，および，USP，ISP，MSPが指定可能。3ビットしかないSFC，DFCも含め，転送は常に32ビット単位で行われる（存在しないビットは0扱い）。

●BKPT

BKPT（BreaK PoinT）は名前が示すようにブレイクポイントに使う命令だが，外部のデバッグハードウェア用であり，X68030では使う意味がない。なお，この命令は68040では事実上サポートされない。

●TRAPcc

ひと揃いの条件つきトラップ命令群。Bccなどの場合と同様，cc部分に任意の分岐条件が入り，その条件が満たされているときに例外を発生する。念のためながら，TRAP #nの条件つき版ではないことに注意したい。位置づけとしては68000からあるTRAPVの一般化であり，ランタイムエラーを検出することを目的とする。発生する例外はTRAPVと共通のベクタ番号7固定だ。

ここで，絶対に例外を発生しないTRAPFにも，それなりの使い道がある。この命令は68020以降ではなんの動作もしないが，68000/10では不当命令になる。68020で拡張/新設されたアドレッシングモードの一部は68000で無理に動かすと誤動作するので，それを利用するプログラムでは事故を防ぐためにプログラム先頭にTRAPFを置いておくとよいだろう。

●CHK，CHK2

ともにデータの値が一定範囲に収まっているかどうかを調べる命令で，範囲外のときには例外（ベクタ番号6）が発生する。TRAPcc同様，ランタイムエラーの検出用だ。CHKは68000からある命令だが，68020以降，32ビットのオペランドがサポートされた。CHK2の使い方はCMP2に準じる。

●CAS，CAS2

CASはCompare And Swap with operandの略。主としてマルチタスク/マルチプロセッサ環境下において，クリティカルなメモリ領域を安全に更新するための命令だ。68000からあるTAS（Test And Set）の延長線上にあると考えてよい。割り込みハンドラを書く場合を除けば，ユーザープログラムで使う機会はあまりないだろう。

●RTE

例外処理関係あたりは68000と68010以降とでずいぶん違っており，例外処理からのリターン命令であるRTEの動作も変更されている。68000でのRTEはスタックトップからSRとPCを取り出すだけの命令だっ

た。68010以降，何種類かの例外スタックフレームフォーマットが規定され，例外の種類によって使い分けられるようになったことから，例外発生時にスタックに積まれたフォーマット番号を調べて動作を切り替えるようになっている。

内蔵キャッシュメモリ

68030は各256バイトの命令用/データ用キャッシュメモリを内蔵している。実行した命令/参照したメモリ内データはキャッシュに記録され，次に同じアドレスが参照されたときにまだキャッシュにその情報が残っていれば（キャッシュにヒットすれば），キャッシュから命令/データが読み出される。内蔵キャッシュのアクセスには外部バスサイクルが必要ないので，その分高速な動作が可能になる。

68030のキャッシュはダイレクトマップ方式と呼ばれる，比較的単純な構成をしている。256バイトのキャッシュメモリは4バイト単位のエントリに分割され，各エントリとアドレスは固定的な1対多の対応にある。詳しくは前の記事を参照してもらいたいのだが，大雑把にいうと，アドレスの最下位1バイトが同じアドレスはキャッシュの同一エントリと対応づけられている。該当エントリが実際にどのアドレスの情報を保持しているかは，アドレス上位24ビットとファンクションコードからなるタグで識別される。タグはエントリ4つごとにひとつなので，キャッシュ1ライン＝4エントリは必ず連続した16バイトの情報を保持する。ただし，エントリ単位でフラグが設けられており，各エントリは独立に有効/無効を決めることができる。

キャッシュの制御はCACRとCAARで行う。CACR/CAARはMOVEC命令で読み書

DOSコールの移動について

68020/30のコプロセッサ命令は，68000では命令の割り当てられていなかったマシンコードFxxx$_H$（F系列未実装命令）に割り当てられている。Human68kではFFxx$_H$をDOSコールに，FExx$_H$をFLOATn.Xのファンクションコールに割り当て，F系列未実装命令を実行することで発生する例外を利用してシステムコールを実現しているから，ぶつかり合いが心配になるところだ。

基本的にはF系列の未実装命令をシステムコールに使うこと自体に問題はない。コプロセッサは0～7のIDで識別され，接続されていないコプロセッサに対してコプロセッサ命令を使うと，68020/68030はコプロセッサとの通信が失敗した時点でそれを認識し，68000同様にF系列未実装命令例外を発行する。コプロセッサ命令の命令フォーマットを見ると，Human68k/FLOATn.Xで使っているのはコプロセッサID7用の命令であり，モトローラがユーザー用に空けてある部分だ。また，68000ファミリのコプロセッサは浮動小数点演算を行う68881/68882とMMUの68851しかなく，68040が浮動小数点演算ユニットとMMUを内蔵してコプロセッサインタフェイスをサポートしなくなったことから，新しいコプロセッサが出てくる可能性自体，低い。

ただ，穴が2つある。

まず，コプロセッサ命令のうち，cpSAVEとcpRESTOREの2つは特権命令であり，ユーザーモードでは使用できない。68020/30は命令の実行に先立って特権レベルをチェックするので，ユーザーモードでこの命令を使うと，未実装命令例外が起こる前に特権違反例外が発生する。DOSコールのうち，FF7F$_H$以下のものはちょうどcpSAVEとcpRESTOREにぶつかるので，ユーザーモードでこれらのコールを使うとうまくない。もっとも，対処は簡単で，特権違反例外経由でもDOSコールが呼び出せるようにすればよい。Human68k V3.0もその方法で穴を塞いでいる。

これだけだったらよかったのだが，もうひとつの穴は致命的だった。cpRESTOREはコプロセッサとの通信に先立って，オペランドで指定されたメモリをアクセスする。したがって，スーパーバイザモードでcpRESTOREとコードがぶつかるDOSコールを発行すると，変なアドレスをアクセスする危険性がある。

表4を見てもらいたい。cpRESTOREと重なるDOSコールのうち，スーパーバイザモードで呼び出すと問題が生じそうな部分をリストアップしてある。たとえば，DOSコールSETPDBは，68030ではID7のコプロセッサに対する，

cpRESTORE　(A0)

として扱われ，68030はA0の指すメモリを読み出そうとし，A0の値によってはバスエラーを引き起こす。運よくバスエラーが起きなかったとしても，A0がI/O空間を指していたりすると，データを読み出したこと自体が問題になるかもしれない。

さらに危険なのは，ポストインクリメントつきのアドレスレジスタ間接形式を適用したcpRESTOREとぶつかるDOSコールFF58$_H$～FF5F$_H$だ。やはり，アドレスレジスタの値によってはバスエラーが起きるのに加え，バスエラーが起きなければ起きないで，アドレスレジスタの値が変わってしまう。頑張れば，ある程度のところまではつじつまを合わせることができそうだが，スーパーバイザスタックを掘り返す，

cpRESTORE　(A7)＋

とぶつかるDOSコールFF5F$_H$(ASSIGN)ばかりはどうしようもないようだ。

結局，メーカーがどう対処したかというと，DOSコールのFF50$_H$～FF7F$_H$をFF80$_H$～FFAF$_H$に移動することにしたらしい（素人考えだと，動かすのはFF50$_H$～FF5F$_H$だけでいいようにも見えるけど，それじゃ駄目なのかな？）。Human68k V3.0ではFF50$_H$～FF7F$_H$もこれまでどおりサポートするが，ユーザーモードでのみ使用可能とし，スーパーバイザモードでの動作は保証されない。今後はFF80$_H$～FFAF$_H$のほうを使うことが推奨されている。既存プログラムのうち，スーパーバイザモードでDOSコールFF50$_H$～FF7F$_H$を発行しているものは，DOSコール番号定義ファイルDOSCALL.MACを差し替えたうえで，再コンパイル/再アセンブルすることになる。

ところが，DOSコールFF80$_H$～FFAF$_H$はHuman68k V3.0でしかサポートされないから，こうして再構成したプログラムは，Human68k V2.03以下では動作しない。もっとも，DOSコールのジャンプテーブルだけの問題なので，DOSコールFF80$_H$～FFAF$_H$が使えるように古いHuman68kを改造するのは簡単だ。リスト1を入力し，～.Rのファイル名で20バイトのファイルにする。これをCONFIG.SYSのPROGRAM行なり，AUTOEXEC.BAT中で実行すれば，古いHuman68kも（この点については）V3.0相当の仕様になる。本当は，デバイスドライバの中でFF80$_H$～FFAF$_H$が使われる可能性に備えて，HUMAN.SYSに直接パッチを当てたほうが安全なのだが，当面はこれでもつだろう。

表4　DOSコールとコプロセッサ命令の衝突

コード	DOSコール	対応するコプロセッサ命令
FF50$_H$	SETPDB	cpRESTORE (a0)
FF51$_H$	GETPDB	cpRESTORE (a1)
FF52$_H$	SETENV	cpRESTORE (a2)
FF53$_H$	GETENV	cpRESTORE (a3)
FF54$_H$	VERIFYG	cpRESTORE (a4)
FF55$_H$	COMMON	cpRESTORE (a5)
FF56$_H$	RENAME	cpRESTORE (a6)
FF57$_H$	FILEDATE	cpRESTORE (a7)
FF58$_H$	MALLOC2	cpRESTORE (a0)＋
FF5A$_H$	MAKETMP	cpRESTORE (a1)＋
FF5B$_H$	NEWFILE	cpRESTORE (a2)＋
FF5C^H	LOCK	cpRESTORE (a3)＋
FF5F^H	ASSIGN	cpRESTORE (a7)＋

リスト1

```
1:          .include            iocscall.mac
2:
3:          lea.l    $1800+$80*4.w,a1
4:          lea.l    $1800+$50*4.w,a2
5:          move.l   #($80-$50)*4-1,d1
6:          IOCS     _B_MEMSET
7:          .dc.w    $ff00
```

リスト2

```
000000  43 F8 1A 00 45 F8 19 40 : EB
000008  22 3C 00 00 00 BF 70 89 : 16
000010  4E 4F FF 00 00 00 00 00 : 9C
000018  00 00 00 00 00 00 00 00 : 00
000020  00 00 00 00 00 00 00 00 : 00
000028  00 00 00 00 00 00 00 00 : 00
000030  00 00 00 00 00 00 00 00 : 00
000038  00 00 00 00 00 00 00 00 : 00
000040  00 00 00 00 00 00 00 00 : 00
000048  00 00 00 00 00 00 00 00 : 00
000050  00 00 00 00 00 00 00 00 : 00
000058  00 00 00 00 00 00 00 00 : 00
000060  00 00 00 00 00 00 00 00 : 00
000068  00 00 00 00 00 00 00 00 : 00
000070  00 00 00 00 00 00 00 00 : 00
000078  00 00 00 00 00 00 00 00 : 00
-----------------------------------------
CKSUM:  B3 83 19 00 45 B7 89 C9 DD52
```

きする。CACRはビットごとに図3のような機能をもつ。

●EI/ED

それぞれ,命令キャッシュ/データキャッシュを使用するかどうかを決め，1のときキャッシュは機能する。

●FI/FD

1にすると命令キャッシュ/データキャッシュを一時的にフリーズ（凍結）する。フリーズしてもキャッシュ自体は有効で，ヒットした場合はキャッシュから命令/データが読み出される。ただ，新たにキャッシュエントリが更新されることはない。この機能は,頻繁に実行/参照する小さなループやデータをキャッシュ上に“常駐”する目的で使用することになるだろう。

●CEI/CED

このビットを1にしたデータをCACRに書き込むことで,命令キャッシュ/データキャッシュの任意の1エントリをクリアすることができる。クリアするエントリはCAARで指定する。CAARにはクリアするエントリに対応するアドレスの代表を入れておく。CEI/CEDは常に0で読み出される。

●CI/CD

このビットを1にしたデータをCACRに書き込むことで,命令キャッシュ/データキャッシュの全エントリがクリアされる。読み出すと常に0が返る。

●IBE/DBE

バースト転送サイクルによる4エントリ単位での充填を行うかどうかを表す。1のとき，バースト転送による4エントリ単位での充填を行う。0なら1エントリ単位で充填する。バースト転送は連続する4ロングワードを1ロングワードあたり通常よりも短いクロック数で転送する機能で,68030では読み込み時，それもキャッシュの充填のときにだけ使われる(68040では書き込みにも使用でき，これを利用して16バイトの高速転送を行うMOVE16なる命令がサポートされている)。

バースト転送によるキャッシュ充填には確率的な賭けの要素がある。つまり，せっかく4エントリ分キャッシュに取り込んでも，いま必要な1エントリ以外が使われるという保証はなく，時間を無駄にしただけで終わってしまう可能性もあるわけだ。しかし，一般にメモリアクセスは局所的に行われるものであり，特にプログラムの実行についてはその傾向が強い。かなり高い確率で次の瞬間には直後の命令が実行されるから，バースト転送による“先読み”は効果的だと考えられる。データキャッシュの場合，メモリを飛び飛びにアクセスするアルゴリズムとの相性は悪いかもしれないが，それは比較的まれなケースであり，通常は高速に先読みできるメリットのほうが大きいだろう。

●WA

メモリに対する書き込み時のデータキャッシュの更新の仕方を決める。0のときは,キャッシュがヒットしたときだけ更新する（ノーライトアロケーションモード）。1のときは必ず更新する（ライトアロケーションモード）。

X68030では常にライトアロケートモードを使うことになるだろう。というのも，キャッシュのタグはファンクションコード，つまり，どのアドレス空間かを含んでおり，X68030のメモリ構成ではメモリ空間と実メモリの対応が重複しているので，同じメモリを指す複数の異なるタグが存在しうるためだ。たとえば，次のプログラムを考えよう。

```
MOVE.L  (WORK,PC),D0
MOVE.L  D1,(WORK)
MOVE.L  (WORK,PC),D2
```

ラベルWORKで指定されたアドレスからの1ロングワードを読んで，書いて，また読んでいるわけだが,読み込みにはPC相対を使い，書き込みは絶対アドレスで行っている。PC相対のアクセスはプログラム空間に対して行われ，絶対アドレスでのアクセスはデータ空間に対して行われるので，両者のキャッシュ上でのタグは異なる。このため，2行目の書き込み時にキャッシュはヒットせず，ノーライトアロケーションモードだと更新されない。したがって，3行目で読み出されるのはD1の値ではなく，その前にD0に読み出したときの古い値となる。ライトアロケーションモードなら，2行目で書き込んだときにもキャッシュが更新され，期待どおりの動作をする。

68000とのプログラム共有

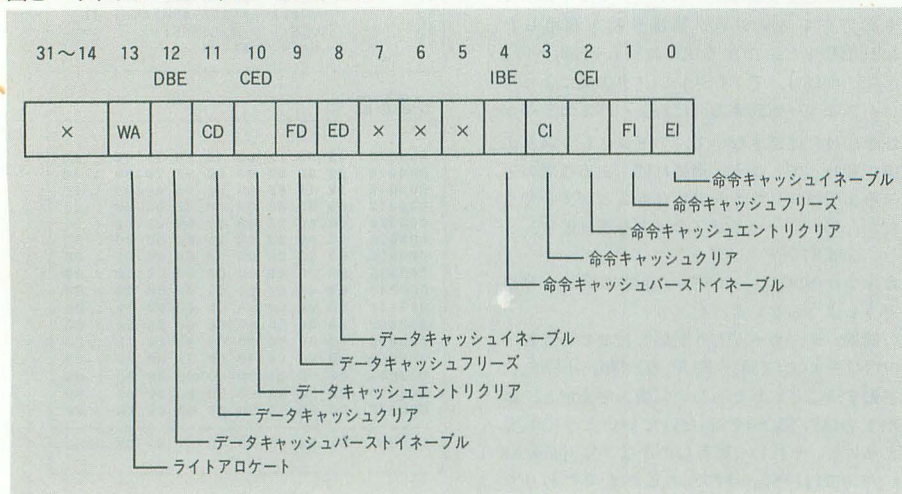
さて,68030の長所を強調してきたわけだが，当面，X68000とX68030は共存していく。基本的に，プログラムはX68000とX68030の両方で問題なく動くように作るべきだ。とりあえず，表向き，僕はそういう立場をとる。そこで，X68000では動いてX68030では動かないプログラムの対処法を兼ねて，X68000/X68030両用プログラムを作るうえでの注意点を挙げておきたい。

●キャッシュ

ユーザーモードで動作するX68000用のプログラムは，自己書き換えの類の技を使ってさえいなければ，X68030でも動く。また，自己書き換えをしているプログラムも命令キャッシュを殺しておけば確実に動く。

命令キャッシュと自己書き換えの相性の悪さは改めて指摘するまでもないと思う。書き換えた部分が命令キャッシュに存在すると，メインメモリの内容とキャッシュ内容の不一致（インコヒーレンシ）が発生する。実際に実行されるのは書き換えたメインメモリ上の命令ではなく，キャッシュに残った古い命令だ。当然，プログラムは期待どおりの動作をしなくなる。互換性以前の問題として，X68030では自己書き換えは禁じ手だ。もっとも，自己書き換え，即，危険というわけではないし，最悪，キャッシュの該当エントリだけをクリアすることもできるわけだが，やるからには自分の責任でやるべき，という考えで，ここでは詳細には立ち入らない。

あと，キャッシュ関係で問題になるとす

図3　キャッシュコントロールレジスタ

ればDMAだろう。DMAはCPUが知らないところでメモリを書き換える。書き換えた結果はキャッシュには反映されないから，やはりメインメモリとの不一致が生じる。68040になると，それを避けるためにバスを監視する機能を持っていたりもするが，68030では純ソフトウェア的につじつまを合わせなければならない。DMAについてはそれなりの配慮が必要だ。ただ，ちゃんと調べたわけではないが，おそらくX68030ではDMAの転送終了時の割り込み処理ルーチンあたりでキャッシュをすべてクリアしているものと思われる。この割り込み処理ルーチンを差し替えたりしなければ，DMAの使用が悪影響をもたらすことは，たぶん，ない。

●ワード境界をまたぐワード/ロングワードアクセス

X68030では動いてX68000では動かないプログラムとしては，拡張された命令/アドレッシングモードを使っているという自明の場合を除くと，ワード/ロングワードデータを奇数アドレスから置いたプログラムが考えられる。まだ，書いていなかったような気がするが，68000では禁止されていたワード境界をまたぐワード/ロングワード単位のデータアクセスが，68020以降では解禁になっている。このことに頼ると，68000の命令セットしか使っていないのに，X68030では動いてX68000では動かないプログラムが簡単に書けてしまう。もっとも，68000でずっとプログラムを作ってきた人であれば，データをワード境界に揃える癖が染みついているだろうから（暴走時を除けばアドレスエラーにはしばらくお目にかかってないでしょ？），いままでどおりにやればいいだけのことだ。

●例外の周辺

スーパーバイザモードで動くプログラムでは，例外周りが危険地帯だ。すでに触れたように，68010以降，例外発生時には68000よりも多くの情報がスタックに積まれるので，例外スタックフレームの深さが違う。また，RTEの動作も変わっている。このあたりが絡むと，X68000とX68030の両方で動作するようにプログラムを作るのはかなり難しくなる。機種を判別して処理を振り分ける必要に迫られるかもしれないし，場合によっては，プログラムを2種類用意したほうが楽な場合も考えられる。

ま，この件に関しては，西川君をモルモットにして，ZMUSICをX68030対応にする過程でのノウハウをみんなして盗めばいいんじゃないか，と，僕はたかをくくっている。

68030の性能を活かすために

表向きの話はすんだので，最後に裏の話を少々。

なんだかんだといって，やっぱりせっかく68030が使えるのだから，その性能を活かしたプログラムを書いてもみたくなる。互換性を保ったほうがいいと頭ではわかっていても，血がそれを許さない。まだ68030対応の開発ツールがない現状では苦しいものもあるが（ROMデバッガはあるけど），追っつけメーカーからの提供があるだろうし，それよりも先にユーザーレベルでの対応が進むだろう。少しぐらい予習しておくのも悪くない。

正直，まだ68030でまともなプログラムを作ったことがあるわけではないのだが，マニュアルをぱらぱらめくっていて目についた点をいくつか挙げておこう。なかには，ほとんど実現不可能で，無理に実現しようとすると“あっちの世界”にトリップしてしまいそうなものもあるが，そこはひとつ，大目にみてもらいたい。

●新設された命令/アドレッシングモードを活用する

いわずもがな。便利な命令/アドレッシングモードはガンガン使う。ポインタのポインタを実現する場合なら，メモリ間接を使えば2命令が1命令になり，また，余分なアドレスレジスタを使わなくてすむようになるから，そのレジスタをほかに転用して……，という具合に，連鎖的な効用が期待できる場合もある。

あと，以前からある命令/アドレッシングモードも実行時間が変わっていたりするので，命令実行時間表は要チェックだ。たとえば，68000ではCLR命令は（相対的に見て）遅い命令だったため，

```
CLR.L D0    →  MOVEQ.L #0,D0
CLR.L -(SP) →  PEA.L 0.W
```

のような置き換えをみんなしてやったわけだ。が，68020以降，データレジスタはCLRでクリアしてもMOVEQで0を転送しても実行時間に差はなく，スタックに0.Lを積むのにPEAを使うとかえって遅い。68000でメモリに対するCLRが遅かったのは，余分なリードサイクルが挟まっていたためだが，その点が改善された結果，こうなっている。

68000では遅くて毛嫌いされたアドレッシングモードである絶対ロング形式も，68020以降ではそんなに遅くはない（ちゃんと32ビットバスにメモリをつないであるという前提）から，無理に避けようとしてごちゃごちゃしたことをやるよりも，素直にすっきり書いたほうがいい場合もあるかもしれない。

ただし，単独の命令の速い/遅いにこだわるのはあまり意味がない。68030では命令が部分的にオーバーラップして実行されるから，あくまで前後の文脈の中で捉える必要がある。

●キャッシュを有効に働かせる

X68000では速度を稼ぐ非常手段（常套手段？）としてループ展開というせこい手が使われた。100回のループを組む代わりにループの中身を100個並べて，ループの終了判定の分を浮かそうという考え方だ。だが，68030には命令キャッシュがあるため，ループを展開するとかえって遅くなるケースも結構出てくるだろう。短いループならキャッシュはヒットしまくりだから，命令をメモリから取り込む必要がないのに対して，ループを展開してしまうとキャッシュはほとんど働かず，全命令をいちいちメモリから読み込まなければならないのだ。仮に，その特定のルーチンではループを展開したほうが速かったとしても，展開したループはキャッシュを無駄に食い潰す。ループを展開した結果が256バイト以上なら，それを実行した時点で以前のキャッシュ結果は上書きされて失われる（一時的にキャッシュを凍結して，更新されないようにすることは可能）。一般に，プログラムを小さくすることと速くすることとは相反するものだが，命令キャッシュの存在下では，両立しうるということを念頭に置いておきたい。

また，一時期にまとめて何度も参照するワークはレジスタに取り出しておくのが常道だが，そのためにレジスタの退避/復帰を伴うようなら，そのままメモリに置いておいてデータキャッシュまかせにしたほうがよい場合もあるだろう。

さらに，68030のキャッシュで採用されているダイレクトマップ方式の癖にも気をつけたい。68030では，最下位バイトが同じアドレスはキャッシュの同じエントリと対応する。このため，ちょうど256の倍数単位でアドレスが離れた複数のメモリに交互にアクセスすると，キャッシュがそのたびに更新され，まるで意味をなさなくなる。頻繁にアクセスするデータは変に離しておかずに，隣接するメモリにまとめておいたほうがよいだろう。この場合，キャッシュのバースト充填も有効に機能するかもしれない。

●アラインメントに気を払う

68020以降，ワード境界をまたいだワード/ロングワードデータアクセスが許されるようになったことで，アラインメントに関する制約は68000よりもむしろ68030のほうが緩い。ただ，68030はこのようなアクセスを2回のバスサイクルに分けて実行するから，性能は低下する。さらに，68030では奇数ワード境界をまたいでロングワード単位のメモリアクセスをすると，やはり2回のバスサイクルに分割されるので遅くなる。

32ビットバスを有効に活かすためには，ロングワードデータは偶数ワード境界に整合しておく（4の倍数のアドレスから格納する）必要がある。特に頻繁にアクセスするデータは，偶数ワード境界にあるかないかで実行速度に差が出てくるだろう。データキャッシュがうまく効けばその差は目立たなくなると考えられるが，整合しておかない理由もない。

ただ，現行のアセンブラ/リンカはワード単位でしか整合をとってくれないから，68030対応の開発ツールが出回るまで，アラインメントにはあまりこだわらないほうがいいかもしれない。どうしてもというのなら，とりあえずリンクしてみて，うまくロングワード境界に並んでいなかったら，手作業でワークの順序を変えたり，スタブを挟んだりして対処することもできるとはいえ，すぐに不要になる手間をかけるのはどうかと思う。

●命令実行のオーバーラップを最大限に利用する

ビョーキ度特A。68030では隣接する命令列の一部を並列に実行することができる。命令やアドレッシングモードごとに，直前の命令とオーバーラップできる時間（ヘッド）と後続の命令とオーバーラップできる時間（テール）が決まっている。実際に，オーバーラップするのは，前の命令のテールと後ろの命令のヘッドのどちらか小さいほうだ。このため，同じ2命令でもどちらを先に実行するかによって，2クロックくらい違ってくる可能性がある(図4)。実際にプログラム全部について考慮するのは不可能だろうが，頻繁に実行されるサブルーチンやループの中身ぐらい，人間命令スケージューラになり切ってみるのもいいかもしれない。僕はやりたくないけどね（でも，やるんだろうな，きっと）。

＊

1月の終わりだったか，2月の頭だったか，そろそろ発表だぞ，という噂が巷を駆けめぐっていたころ，久しぶりに覗いた某書店では68030のマニュアルが平積みされていた。うんうん，みんなちゃんと勉強しているね，と勝手に解釈してそこはかとない安心感を覚えつつも，つくづく妙なユーザー層だよなー，と苦笑する。見ているそばからひとりのにーちゃんが，迷うことなく山の上から2冊目を引き抜いて，まっすぐレジへと向かっていった。

図4　命令実行のオーバーラップ

プロセッサの判別法

プログラムによっては，X68000上で動作しているのかX68030上なのかを判別して，処理を振り分ける場合も出てくるだろう。もっとも，X68030のどこかのシステムポートを読めば，必要な情報は得られるはずだし，ひょっとすると，それもんのIOCSコールが新設されているかもしれない。それでもなお，ソフトウェア的にプロセッサを判別するのは，ちょっとした頭の体操になる面白い問題だ。X68000にアクセラレータを差したときに必要になるかもしれないし。で，少し考えてみた。

まず，最初に思い浮かんだのは，適当な例外を発生させてみて，形成されたスタックフレームで判別するという方法。たとえば，不当命令例外が発生したとき，68010以降なら例外処理ルーチンに制御が移った時点で，スタックフレームのオフセット6～9に例外スタックフレーム番号とベクタオフセットが積まれる。68000では積まれない。これで，68000と68010以上が判別できる。似たような案としては，新設された命令を実行してみて，不当命令例外が起きるかどうかを見る方法もある。こちらなら，68000/10/20/30/40を厳密に区別することも可能だ。ただ，どちらの方法も例外処理ルーチンを用意したうえで，例外ベクタを書き換える必要があり，少々大げさすぎるきらいがある。

次に考えたのは，MOVE from SRが68010以降では特権命令になったことを利用する方法。本文でも触れたように，X68030上でユーザーモードのままこの命令を使うと，特権違反例外ルーチンがこっそりMOVE from CCRに置き換える。だから，MOVE from SRが，実行後もまだMOVE from SRのままか，MOVE from CCRに化けたかを調べることにより，68000か68010以上なのかがわかる。この方法は十分手軽で，実用になりそうだ。しかし，ユーザーモードでしか利用できない，特権違反例外ルーチンの動作に頼っている，ただ一度しか実行できない，などの問題もある。

最終案がリスト3だ。難しく考えすぎて損したという話もあるが，68020以降でサポートされたスケールファクタが68000/10では無視されることを利用している。肝は，

```
moveq.l  #1,d0
move.b   (table-1,pc,d0*2),d0
```

の部分。68000/10では，

```
moveq.l  #1,d0
move.b   table-1(pc,d0),d0
```

と解釈されるので，

table－1＋1＝table

の1バイトが読み出される。68020以降では，スケーリングが働いて，

table－1＋1×2＝table＋1

が読み出される。だから，table以降の2バイトに違う値を入れておけば，両者が区別できる。

リスト3では68000/10なら0，68020/30なら1とし，これをそのままプログラムの終了コードとしている。

縁起ものなのでダンプリストも用意した。リスト4の12バイトを適当な方法で入力し，拡張子を.Rにしたファイル名（ここでは仮にIS68000.R)でセーブしておく。あとは，リスト5のバッチファイルのようにプログラムの終了コードで処理を振り分ければよい。

リスト3

```
 1:          .include doscall.mac
 2:
 3:          moveq.l #1,d0
 4:          .dc.w   $103b,$0205
 5: *        move.b  (table-1,pc,d0*2),d0
 6:          move.w  d0,-(sp)
 7:          DOS     _EXIT2
 8: table:   .dc.b   0,1
 9:
10:          .end
```

リスト4

```
000000  70 01 10 3B 02 05 3F 00 : 02
000008  FF 4C 00 01 00 00 00 00 : 4C
000010  00 00 00 00 00 00 00 00 : 00
000018  00 00 00 00 00 00 00 00 : 00
000020  00 00 00 00 00 00 00 00 : 00
000028  00 00 00 00 00 00 00 00 : 00
000030  00 00 00 00 00 00 00 00 : 00
000038  00 00 00 00 00 00 00 00 : 00
000040  00 00 00 00 00 00 00 00 : 00
000048  00 00 00 00 00 00 00 00 : 00
000050  00 00 00 00 00 00 00 00 : 00
000058  00 00 00 00 00 00 00 00 : 00
000060  00 00 00 00 00 00 00 00 : 00
000068  00 00 00 00 00 00 00 00 : 00
000070  00 00 00 00 00 00 00 00 : 00
000078  00 00 00 00 00 00 00 00 : 00
-----------------------------------------
CKSUM:  6F 4D 10 3C 02 05 3F 00 9B21
```

リスト5

```
echo off
is68000
if errorlevel 1 goto not68000
echo 68000です
goto done
:not68000
echo 68000ではありません
:done
```

数値演算コプロセッサを使う

Fの美学

X68030

Taki Yasushi 瀧 康史

CPUが32ビットになってうれしいことのひとつは68881/2がコプロセッサとして接続されることです。ここではさっそく68882を使ってプログラムを組んでみました。

数値演算プロセッサからコプロセッサへ

fsin.x fp0,fp1

これがやりたかった。

68000CPUの良心。確かに86系CPUに比べたら，ずっと良心的ではあった。486と68000のどちらを選ぶ？　といわれたら，私は自信を持って68000といえる。いくら486で，いろいろな欠点が解決されたといえ，速いだけのCPUなら使いたくはない。[*1]

ところが，実数演算に関しては，X68000側では，相当うまいことをしないと，8086＋8087よりも劣ってしまう。FLOATn.xのシステムを使ってるとそれはなおさらで，これはもう最悪の極地としかいえない。

一部のヘビーユーザー，パワーユーザーはフリーウェアなどを駆使してうまく実数計算を行った。実際，私の怪しいX68000XVI（30MHz＋68881），そして私が改良したFPPP.X[*2]においての環境では，Whetstone値（ver.2.0B）を900近くまで到達させることができた[*3]。しかしだ。どんなにうまくやってもその「数値演算プロセッサ」のオーバーヘッドはかわしようがなかった。

本来，プロセッサがハードで自動的に行うものを，いくらうまくやったとしても，ソフトでこれを行うにはやはり重いことだ。結果的には，プログラム中に実数演算の処理を含めば含むほど，全実行時間中の演算プロセッサへのオーバーヘッドにかかっている，本来ならなくてすむべきはずの時間の割合が徐々に増えているので，やはりこれは解決せねばならないひとつの項目となっていた。

また，開発においても，大きなネックがあった。このFPPP.Xが比較的マイナーであった点もある[*4]。しかし，命令は基本的に複数行に展開するしかないため，結果的に，1行が複数行になってしまい，アセンブラがエラーを出した行と，自分のソースの行が等しくなくなってしまうという危険な症状を持っていたのである。

ソースが大きくなればなるほど，自分の書いたソースとアセンブラのエラーメッセージの行のギャップが広くなるため，Micro Emacsなどのマルチウィンドウのエディタを駆使して，ひとつに自分の書いたソース，FPPP.Xが吐き出したソース，そして，アセンブラが出したエラーメッセージと，嫌になるような環境を作りプログラムせねばならなかった。ましてやデバッガを使うときになるとどうなるか……。

プログラマが死ねばそれですむ話だが，すべてのプログラマにユーザーのために「死ね！」というのはあまりに酷い。

ところが時代は変わった。68030になったおかげで，68881/2は数値演算プロセッサからコプロセッサに昇格した。もう，ややこしいデバッグはしなくてよい。デバッガもコプロセッサ命令をあたかも普通の命令として実行してくれる（当たり前の話だが)。さあ，全国のプログラマよ死んでくれ……。

＊1　86系のCPUはおそらく互換のためだろうが，いつまでたっても速いだけのCPUという感を拭い去ることができない。もしスピードを除いたアーキテクチャを見るなら，あれがつい最近になってできたCPUだとは，思えない節がときどきある。ちなみに私はV60，V70，V80が好きである。

＊2　ソース中に，Fで始まる実数演算命令が入っていた場合，X68000＋68881のシステムで動かすことができるように，うまくマクロなどに置き換えるフィルタプログラム。

GCCでは，-m68881オプションを指定すると，float，doubleなどの計算を，直接F系列の命令に出力する。これをFPPP.Xにかけ，うまいことをやると，直接mc68881を叩いた，実数演算が高速に行えるバイナリを出力することができたのである。

参考までに，これを使わないXVI16MHz＋FLOAT2.Xの組み合わせでは，Whetstoneが100ちょっとしかいかない。前述した私のXVIはノーマルXVIの2倍は速くなっていないので，900という数字は(SUNのmath-68881という特殊なインクルードファイルを使っていたにせよ)，どれほど，このプログラム＋68881で稼いでいるかがわかるだろう。なお，Whetstoneとは，実数演算バリバリのベンチマークだと思ってくれればほぼ間違いない。

＊3　この値はFM TOWNSの386＋387を遥かに凌いでいる。

＊4　複数の人が作ったいろいろなバージョンがあるようだが，いったいこれほどまでに便利なソフトを，演算プロセッサを持ってる人間で有効に利用してない人間が何人いるのだろう……。

実数演算の価値

実数演算が速くなるといっても実際どのようなところで実数演算が役に立つか？一般的なユーザーサイドからなにも変わらなければさして意味はないので，あえて考えてみる。

実のところ一般的なアプリケーションユーザーの場合，ほとんど実数演算は不必要となる。たとえば，見方を変えればSX-WINDOWの上で，ワープロを使う場合，通信ソフトを使う場合，はてはハードディスクやMOなどのファイル管理を行う場合，これらすべてに実数演算は必要ないといっていい。リレーショナルデータベースや，Lotus1-2-3などに知られる表計算システムが，SX上に仮に移植されたとしても，実数演算が出てくるのは珍しいケースで，式を整数化してしまって高速化したほうが有効かもしれない。

見る人が見れば一見有効そうな，Lhaの圧縮，展開などもまったく関係ないといっていいし，遅いと不評のZPCNVの計算も，強いていえばロガリズムの計算を地道にやるとしたら有効になるぐらいで，メーカー推奨どおり，所詮は4ビットPCMなのだから，倍算してテーブルを持ってしまったほうが速いし合理的だ(すでにやっている)。

そうなると，実数演算自体どのようなことで役に立つか？　実数演算は結果的に整数化できてしまったら，そちらのほうがスピードが速いので，逆にいえば整数化できない場合のみ，コプロセッサの価値がでる

ともいえる。

そこで整数化できないプログラムにはどういうものがあるか考えてみることにする。あえていえば，X68000は実数演算が苦手なマシンだった。いや，正確には実数演算が苦手なマシンに見えた*5。だからみんな実数演算を敬遠した。整数化すれば実行速度は速くなるがその分，プログラマの負担がかなり大きくなる。

中，高生にもわかるように比喩的に表現するとすれば，実数演算を整数化するのは，Σの計算をループで展開するのではなく，できるだけ，人間にとってではなく，コンピュータにとって演算が簡単になるように式を変形するようなものだ。

これだけ苦労しても，整数化できないプログラムといえば，まずレイトレーシングが挙げられる。いつか中野氏となんとか整数化およびテーブル化できないだろうか？と一緒に考えたことがあったが，初等関数などの引数に実数が入った場合など，こういうケースでテーブル化したら，出来上がりがどうなるかは容易に想像できよう*6。よってこのようなときはコプロセッサが大いに役に立つ。

レイトレーシングはまだまだ一般的ではないと思うので，もう少し一般的なことを考えてみる。それは最近有名になってきた，JPEGなどの不可逆圧縮である。「JPEGもそろそろかなり枯れた技術なので，もうちょっとマトモに考えれば，かなり質のよい不可逆な画像が得られるのではないか」というのは，PICのアルゴリズムの柳沢氏*7の言葉であるが，このような2次元グラフィックの非可逆圧縮の場合，コプロセッサが功を奏する。2種類のバイナリを持てば，コプロセッサなしと，ありでは実に100倍以上は速度に違いが現れよう。

非可逆圧縮ついでに，同様なことが音声データの圧縮についてもいえるのをご存じだろうか？　おそらくMDも同じことをやってるのだろうが，FFTなどを使って可聴域の波形だけを算出し，その部分だけに圧縮をかけるとする。この方法は，ロックはともかく，クラシックなどのフルトゥッティを録音すると腐るので好きではないのだが，X68000のAD PCMの場合，AD PCMの上限から8kHz以上は切り捨て可能だし，4ビットなのだから，ある一定の周波数以下は意味を成さない*8ため，これもまた切り捨てられる。

さらに，AD PCMデータの大半は，上から下まで混ざり切った音が録音されてるのではなくて，あるひとつの音，特にバスドラの音，シンバルの音，タムの音，聞けばわかるが，ある一定の周波数帯に固執する音ばかりである。筆者が昔，この方法で10MHzのX68000演算プロセッサなしでやった場合，かなりクオリティが高いものの，120KバイトあったAD PCMファイルが，30Kバイトほどまで圧縮できた。もっとも，圧縮時間が36時間以上かかったので実用的ではなかったのであるが，これがコプロセッサを積んだ68030で，Cではなくアセンブラで緻密に書かれたとしたら，Lha並みのスピードにはなるだろう*9。とすれば，Oh!X LIVEにPCMデータを掲載できるかもしれない。

圧縮だけでなく，画像関係や音関係のフィルタ，エフェクタを作る場合，実数演算はかなり役に立つ。MATIERなどもコプロセッサに対応して緻密に考えれば，行列的な写像変形は速くならないかもしれないが，結果的にエフェクト関係は比べものにならないほど速くなるだろう。

また，PCMの音をいちから作るという文化がいままでX68000ユーザーにはなかったのか，録音した音，もしくは蚊が刺した程度の変調ばかりだったが，うまくコプロセッサをいかせばかなり面白いエフェクトができる。ループをいかした倍音生成，D70で有名なDLM，FM音源と同じモジュレーション，さしずめ4ビットPCMなのが非常に痛いところだが，考えられることは山ほどあるだろう。

今回，図もなんにも描かず，最適化や，パイプライン，キャッシュの効能*10をも考えず，いきあたりバッタリでコプロセッサの威力を測ろうと直方体への3次元マッピングを行うプログラムを作ってみた。適当に作った割には，それなりのスピードを維持することができた。サンプルプログラムの詳細はあとで触れるとしても，1面描くのに一瞬とはいかなかったが，10秒もかからなかったので，コプロセッサがどれだけ頑張ってくれてるかがわかるだろう。

＊5　まさかWhetstoneで900近い数字を出すと思わなかった。この値から，「愛」があれば決して実数計算のそれほど遅いマシンではなかったことがわかる……。

＊6　しかし，テーブル化することによって，実行速度を稼げるとしたら，プレビュアとして役に立つとは思わないか？

＊7　彼はいま可逆圧縮のPIC2を手がけていて，非可逆圧縮はポリシーにあわないそうで，やる気はないそうだ。

＊8　聴こえる聴こえないという個人の能力に影響するのではなく，もともとなってないのだからしようがない。

＊9　実用的なスピードにはなるだろう。

＊10　まだ未知数であるためという理由があるが，おいおい，本誌でも明かしていくことになるだろう。

68881/2とはどういうものなのか？

本当は初めにこれを述べるべきだったと思うのだが，こういうタイプのものは，結果的に一般的なユーザーが，どのような利益を得るかが，まず重要視されなくてはならないポイントであるというのが私の主義なので，あえて結果から先に持ち出させてもらった。

さて，68Kシリーズには大きく分けて，2つのコプロセッサがあるが，そのひとつは68851のページ式メモリ管理ユニット，つまりPMMUと，今回取り上げる68881/2の浮動小数点演算コプロセッサ，FPCPである。

68881と68882の違いは残念ながら，時間不足からきた資料不足で，イマイチはっきりとした詳細がわからなかったが，68882は68881にピンコンパチ，命令アッパーコンパチであり，そして高速であることは間違いないようだ。

おそらく，内部でシコシコやってるのだろうが，68882はひょっとしたらパイプラインをやってるのかもしれない。資料が揃い，ノウハウが揃ったら，これらはおいおい発表していくとして，ここでは，ぜいたくな人への68882，貧乏な人への68881という程度にとどめておこう。XVIのときと違って，X68030はこの角度で入れてくださいとシールが貼ってあるので，間違える人は少ないと思うから，自信のある人は中古でも新品でも秋葉原あたりを出歩いて，どんどん数値演算コプロセッサを探してできるだけ安く購入し，差し込んでみてエンジョイしてほしい。ちなみに，秋葉原で68881は相場，16MHzが13000円ぐらい，25MHzが2万弱ぐらい。68882は1万～1万5000千円増しと見ていい。仮にそれ以上取られたとしたら，ぼられたと思ってほしい。もっとも，5万円程度で実数演算が100倍近く速くなれば，考え方によっては安いものかもしれないが……。

MPUにコプロセッサがついた場合，ハードウェア設計者はそれにともなった資料，知識が必要になる。これは，コプロセッサインタフェイスとそのプロトコルに沿っていろいろやるわけだが，68030にコプロセッサとして接続された場合，これらは意識せずにMPUに単に浮動小数点演算のための命令が増えたと考えるだけでよい。68000

につけたときは，本来，68020/30と演算プロセッサ68881/2とのあいだで行われる送受信をうまくバス接続したときにエミュレートする必要があったのだが，コプロセッサになれば，ただアセンブラのなかに，そのまま記述すればよいだけで，80ビットの8本の強力なデータレジスタもどんどん使うことができる。先に述べたFPPP.Xで旧機種も同じようなことができたのだが，これはソースの上だけの話で，デバッガを起動したときにこの2者の違いは軒並み表れてくる。デバッガを起動したときも，68030ではほかのレジスタと同じように，FPレジスタの内容が（いうまでもないが）実数表記で現れる。1命令は1命令だし，XVIで使ってたときのように，いつのまにやら68881が暴走していて，CIRに直接リセットをかけたなんてこともうしなくてよい(当たり前だが)。

どうも脱線してしまうので，話を強制的に元に戻す。

そのコプロセッサはメインプロセッサと常に同期を取りながら動作する。コプロセッサの命令はコプロセッサIDの都合上，F2??で始まる命令なので，メインプロセッサは命令実行中にこのF2??を見つけると指定のプロトコルに従って，命令をコプロセッサ側に渡し，結果をもらってから次の命令に進む。

このとき，デスティネーションがメインプロセッサ側のレジスタであったり，メモリであった場合，結果的にメインプロセッサは結果を待たず，先に進むため，その一瞬は並列動作する。

この指定のプロトコルというのは，もちろんアドレッシングモードによっても違うが，主に命令の種類によって違い，これが去年の9月号でやったOPコードというやつである。これらはすべて自動的にメインプロセッサが判断して，コプロセッサ側に引き渡す。

メインプロセッサ側には汎用的なコプロセッサにデータを引き渡すために，cpGENという命令があるが，68881/2の場合は特にこれらは意識しなくてよい。

68881/2でサポートされる命令のアセンブラシンタックスはメインプロセッサに準ずる。68881/2は内部的にはすべて.X形式，すなわち，拡張実数型(12バイト)で扱われているが，メインプロセッサはそうではないため，オペランドサイズは，B，W，Lのほかに S，D，X，Pを指定できる。これら S，D，X，Pは，

S：実数型(4バイト)
D：倍精度実数型(8バイト)
X：拡張実数型 (12バイト)
P：パックドデシマル型

であり，SはC言語などの，FLOATにそのまま，DはC言語などの，DOUBLEにそのまま当てはまる。

先に述べたとおり，内部ではすべて.X形式で処理されているため，このオペランドサイズは，ソースがメインプロセッサ側だった場合にはソースの，デスティネーションがメインプロセッサ側だったら，デスティネーションのオペランドサイズを表していることに注意してほしい。すなわち，

FADD.L D0,FP0

の.LはD0のオペランドサイズを意味するが，

FMOVE.L FP0,D0

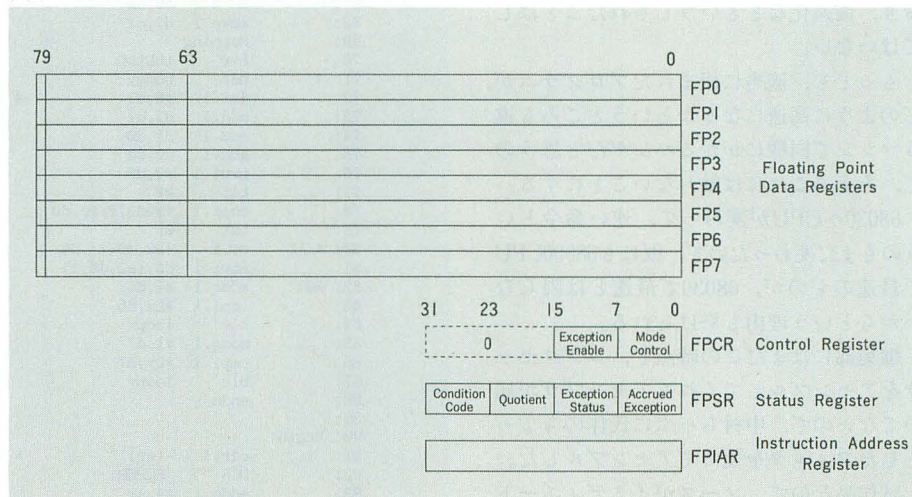
の.Lはデドスティネーション側のD0のオペランドサイズを表すというのである。

精度の拡張や丸めなどはすべてコプロセッサ側が自動で行う。サンプルプログラムは画面に出力するものなので，計算後すべてメインプロセッサ側のデータレジスタに値を丸めて転送するが，これをもし，そのまま拡張精度型の実数で吐き出させたい場合は値をD0-D2までに入れたり，(A0)とかの表記にしてデータをアドレスレジスタ相対にしてメモリに押し込めばよい。

ただ，D0-D2などの表記は，X68030対応のアセンブラがまだ編集部に届いてないため，どうなるかイマイチ不確定ではあるが，おそらくそんなところだろう。

気がついた方もいると思うが，IEEEの拡張精度，パックドデシマルは96ビットなのに，68881/2の内部レジスタは80ビットという点である。これは，拡張精度実数，パックドデシマルの規格上，常に0のポイントがあるため，それらを省略して内部では80ビットで扱われるのである。

68881/2はこれ以外にもFPCR，FPSR，FPIARなどの機能的なレジスタを持っている。

図1 MC68881/68882プログラミングモデル

リスト

```
 1: *           必殺のなんにも考えてないプログラム
 2: *
 3: *                       ぱ～ぃ たき。
 4: *
 5: *           なんとなくプログラムにバグがあるような気がするがまぁいいだろう・・・
 6: *                                                     よくないってば・・・
 7:
 8:             .include        doscall.mac
 9:             .include        iocscall.mac
10:
11:             .text
12:
13: CRT         equ     24      * IOCS _CRTMOD の値
14:
15: Wd          equ     256     * 直方体の横の長さ
16: Hd          equ     256     * 直方体の縦の長さ
17: Dd          equ     256     * 直方体の奥行き
18:
19: Lx          equ     256     * データマップ上のX軸方向の大きさ
20: Ly          equ     256     * データマップ上のY軸方向の大きさ
21:
22: Xmv         equ     320     * X軸の移動値
23: Ymv         equ     384     * Y軸の移動値
24:
25: Center      macro
26:
27:             mulu.w  #Wd/Lx,d0
```

FPCRはコントロールレジスタで，コプロセッサ例外的なイネーブル/ディセーブルの制御，計算結果の丸めの指定を行うものであり，FPSRは演算エラー(無限大に起因するものなど)や結果のフラグ，除算命令の商などまで保存される。FPIARは実行された最後の浮動小数点命令のアドレスを保持するためのもので，割り込みによって中断されたときに，どの場所で再開するかを判断するためのものなので，システムユーザーはあまり意識していじってもらっては困るものである。

サンプルプログラム

サンプルのプログラムは，6万色モードの256×256の範囲の画面を直方体の3面へ，マッピングするプログラムである。

画像関係のエフェクト処理は，私にはノウハウはなかったので，ありのままの式から，そのままコプロセッサに与えるプログラムにしてしまったために，先に述べたとおり，最適化などというしゃれたことはしてはいない。

もっとも，適当に組まれたプログラムが，どのように高速になるかというところも違うマシンで同様にかかるハンデだと思うので，あえてこれには触れないことにする。

68030へCPUが変わって，速い命令というのもまた変わったので，仮にも68000CPUで最速のものが，68030で最速とは限らないからという理由も挙げられる。

編集部にはまだこの時点で，コプロセッサをアセンブルしてくれるアセンブラが届いてないので，中村ちゃぷに氏作のちょっとしたフィルタを使ってアセンブルした。

試作品なので，ソースがイミディエートの命令はアセンブルできず，その部分は，直接バイナリをDCで展開したため，汚くなってしまった。コメントとしてその部分には必ず同じ意味を成す命令を記述したので，掲載されたプログラムが読者の手に渡り，もしくは，完成されたアセンブラが読者の手に渡ったときには，そちらを入力したほうが都合がいいだろう（そのまま入力しても実行できるが)。

ただし，ソースは完全にX68030のものなので，従来のユーザーはこのソースをそのまま通すことはできない。

ベンチマークといえば，できるだけ公平な立場で行わねばならないこともあったが，そもそも，68000にはコプロセッサ命令がないことから，先に述べたフリーソフトウェアの，FPPP.Xを使ってアセンブルして

```
 28:          mulu.w  #Hd/Ly,d1
 29:          sub.l   #Hd,d1
 30:          neg.l   d1
 31:          move.l  #Dd,d2
 32:          endm
 33:
 34: High     macro
 35:          mulu.w  #Wd/Lx,d0
 36:          mulu.w  #Dd/Ly,d1
 37:          move.l  d1,d2
 38:          move.l  #Hd,d1
 39:          endm
 40:
 41: Side     macro
 42:          mulu.w  #Dd/Lx,d0
 43:          sub.l   #Dd,d0
 44:          neg.l   d0
 45:          move.l  d0,d2
 46:          mulu.w  #Hd/Ly,d1
 47:          sub.l   #Hd,d1
 48:          neg.l   d1
 49:          move.l  #0,d0   *wd
 50:          endm
 51:
 52: Map      macro   Mapping           * こんなもんマクロにするなよ・・・
 53:          local   loopy
 54:          local   loopx
 55:          lea.l   $C00000,a2
 56:          lea.l   GrBuf,a0
 57:          move.l  #0,d7             * d7=y
 58: loopy:   move.l  #0,d6             * d6=x
 59: loopx:   move.l  d7,d1
 60:          move.l  d6,d0
 61:
 62:          lsl.l   #8,d1             * (y*512+x)*2
 63:          add.l   d1,d1
 64:          add.l   d1,d0
 65:          add.l   d0,d0
 66:          move.l  (a0,d0.l),d5
 67:          move.l  d6,d0
 68:          move.l  d7,d1
 69:          Mapping
 70:          bsr     thDtwD
 71: *        bsr     Dummy
 72:          lsl.l   #8,d1             * (y*512+x)*2
 73:          add.l   d1,d1
 74:          add.l   d1,d0
 75:          add.l   d0,d0
 76:          cmpi.l  #0,d0
 77:          blt     @f
 78:          cmpi.l  #$0007fffc,d0
 79:          bgt     @f
 80: *        or.l    (a2,d0.l),d5
 81:          move.l  d5,(a2,d0.l)
 82: @@:      addq.l  #1,d6
 83:          cmpi.l  #Lx,d6
 84:          ble     loopx
 85:          addq.l  #1,d7
 86:          cmpi.l  #Ly,d7
 87:          ble     loopy
 88:          endm
 89:
 90: Begin:
 91:          clr.l   -(sp)             * Superへ
 92:          DOS     _SUPER
 93:          addq.l  #4,sp
 94:
 95:          lea.l   $C00000,A0
 96:          lea.l   GrBuf,A1
 97: @@:      move.l  (a0)+,(a1)+
 98:          cmpa.l  #$c80000,a0
 99:          blt     @b
100:
101:          move.l  #CRT,d1
102:          IOCS    _CRTMOD
103:          IOCS    _G_CLR_ON
104: *        IOCS    _B_KEYINP         * キー待ち
105: *@@:     bra     @b
106: *        bra     NormalDisp        * ここに飛ぶなら、キー待ちした方がいいよん。
107:
108: *        座標変換のデータ作成部（重いけど一度だけだ！ごりごりいけ！）
109:
110:          fmovecr #0,fp0
111:          fmove.x #0,fp2                        * fp2=Xan
112:          fmul.x  fp0,fp2                       * fp2=Xan(rad)
113:          fsin.x  fp2                           * fp2=sin Xan(rad)
114:
115:          fmove.x #0.1666666666666,fp3              * fp3=Yan
116:          fmul.x  fp0,fp3                       * fp3=Yan(rad)
117:          fmove.x fp3,fp4                       * fp4=Yan(rad)
118:          fcos.x  fp3                           * fp3=cos Yan(rad)
119:          fsin.x  fp4                           * fp4=sin Yan(rad)
120:
121:          fmove.x #0.1666666666666,fp5              * fp5=Zan
122:          fmul.x  fp0,fp5                       * fp5=Zan(rad)
123:          fmove.x fp5,fp6                       * fp6=Zan(rad)
```

マッピング実行例

みた。

このプログラムはアドレスレジスタをフルには使ってないため，A5あたりをCIRアドレスに設定し，CIRとのデータの取引をアドレスレジスタ相対にしたほうが速いため，旧機種の最高のスピードとはいえないのだが，なにせ環境が揃ってない点から許していただきたいと思う（経験から実行スピードは5％ぐらい速くなると思う）。ルール違反ではあるが，FPPP.Xの作者には頑張っていただきたい。

FがX68030をどう救うか？

ここまで読んでくれた読者には，実数演算がどのようなところで役に立つかは，もうおわかりであると思う。

真金取っ主（いい変換だ）のまっとうなマシンはコプロセッサがついてるはずだが，できるなら，X68030にもコプロセッサを標準装備していただきたかった。

コプロセッサを使う命令がループの中に入るときには，なにやらプログラムキャッシュが効いてないようだとか，不確定に測定された結果はほかにもあるが，なにせまだいまの段階では情報が揃ってない点がうらめしい。

Fから始まる命令がX68030の実数演算環境に対してどのような効果を示すかどうかは，いまは未知数だしどう転ぶかもわからない。強いていえば，Fから始まる命令は実数演算において，有利に進めるポイントでもあるし，あえて避けてきた実数演算の環境がどのようになるかも，注目したいところである。

なぜなら，それが「Fの美学」であって，同時に「Fへの探求」にもなりうるからである。

表1　マッピングの実行時間(秒)

X68030＋68882(25MHz)	15.3
X68000XVI＋68881直接駆動(16.6MHz)	41.1
X68000XVI＋68881直接駆動(30MHz)	25.0

```
124:            fsin.x  fp5                     * fp5=sin Zan(rad)
125:            fcos.x  fp6                     * fp6=cos Zan(rad)
126:
127:            Map     Center
128:            Map     High
129:            Map     Side
130:            DOS     _EXIT
131:
132: Dummy:     subq.l  #1,d0
133:            subq.l  #1,d1
134:            rts
135:
136: thDtwD:            * これがごりごり呼ばれるのは、やはり重そうだ。おもいぞ～
137: *
138: *          レジスタD0．1に引き数X
139: *          レジスタD1．1に引き数Y
140: *
141: *          レジスタD0．1に戻り値Px
142: *          レジスタD1．1に戻り値Py
143:
144: *          Px = X*fp3*fp6+Y*(fp2*fp4*fp6-fp3*fp5)+Z*(fp3*fp4*fp6+fp2*fp5) + Xmv
145: *          Py =-X*fp3*fp5-Y*(fp2*fp4*fp5+fp3*fp6)-Z*(fp3*fp4*fp2-fp2*fp6) + Ymv
146: *          Pz =-X*fp4+Y*fp2*XF+Z*fp3*fp3                 ?
147:
148: *          なにも考えていないので参考にしないこと。
149:
150:            move.l  d0,d3           * シードとなるこいつらをとっておく
151:            move.l  d1,d4
152:
153:            fmove.l d3,fp0          * fp0=x
154:            fmul.x  fp3,fp0         * fp0=fp0*fp3
155:            fmul.x  fp6,fp0         * fp0=X*fp3*fp6
156:            fmove.x fp2,fp1
157:            fmul.x  fp4,fp1
158:            fmul.x  fp6,fp1         * fp1=fp2*fp4*fp6
159:            fmove.x fp3,fp7
160:            fmul.x  fp5,fp7         * fp7=fp3*fp5
161:            fsub.x  fp7,fp1         * fp1=(fp2*fp4*fp6-fp3*fp5)
162:            fmul.l  d4,fp1          * Y*(fp2*fp4*fp6-fp3*fp5)
163:            fadd.x  fp1,fp0         * X*fp3*fp6+Y*(fp2*fp4*fp6-fp3*fp5)     * fp1,fp7があき
164:            fmove.x fp3,fp7
165:            fmul.x  fp4,fp7
166:            fmul.x  fp6,fp7         * fp3*fp4*fp6
167:            fmove.x fp2,fp1
168:            fmul.x  fp5,fp1         * fp2*fp5
169:            fadd.x  fp7,fp1         * (fp3*fp4*fp6+fp2*fp5)
170:            fmul.l  d2,fp1          * Z*(fp3*fp4*fp6+fp2*fp5)
171:            fadd.x  fp1,fp0         * X*fp3*fp6+Y*(fp2*fp4*fp6-fp3*fp5)+Z*(fp3*fp4*fp6+fp2*fp5)
172:            fmove.l fp0,d0          * fp0,1,7があき
173:            add.l   #Xmv,d0
174:
175:            fmove.l d3,fp0
176:            fneg.x  fp0
177:            fmul.x  fp3,fp0
178:            fmul.x  fp5,fp0         *-X*fp3*fp5
179:            fmove.x fp2,fp1
180:            fmul.x  fp4,fp1
181:            fmul.x  fp5,fp1         * fp2*fp4*fp5
182:            fmove.x fp3,fp7
183:            fmul.x  fp6,fp7         * fp3*fp6
184:            fadd.x  fp7,fp1         * (fp2*fp4*fp5+fp3*fp6)
185:            fmul.l  d4,fp1          * Y*(fp2*fp4*fp5-fp3*fp6)
186:            fsub.x  fp1,fp0         *-X*fp3*fp6-Y*(fp2*fp4*fp5-fp3*fp6)     * fp1,fp7があき
187:            fmove.x fp3,fp7
188:            fmul.x  fp4,fp7
189:            fmul.x  fp2,fp7         * fp3*fp4*fp2
190:            fmove.x fp2,fp1
191:            fmul.x  fp6,fp1         * fp2*fp6
192:            fadd.x  fp1,fp7         * (fp3*fp4*fp6-fp2*fp5)
193:            fmul.l  d2,fp7          * Z*(fp3*fp4*fp6+fp2*fp5)
194:            fneg.x  fp7             *-Z*(fp3*fp4*fp6+fp2*fp5)
195:            fadd.x  fp7,fp0         *-X*fp3*fp5-Y*(fp2*fp4*fp5+fp3*fp6)-Z*(fp3*fp4*fp2-fp2*fp6)
196:            fmove.l fp0,d1
197:            add.l   #Ymv,d1
198:            rts
199:
200: NormalDisp:                    * こうすると、画面を裏画面からそのまま転送。
201:                                * キャッシュはオンでね！
202:            lea.l   GrBuf,A0
203:            lea.l   $C00000,A1
204:            move.l  #$7ff,d0
205: @@:        .rept   64
206:            move.l  (a0)+,(a1)+
207:            .endm
208:            dbra    d0,@b
209:
210:            DOS     _EXIT
211:
212:            .even
213:            .bss
214:
215: GrBuf:
216:            .ds.b   $80000          * ど～だ～86マシンにはできめぇ、おうん
217:
218:            .end
```

特別寄稿 X68030によせて

祝一平の「終わりのない物語」

そうだな，もう，だいぶ昔のことになってしまったけどね。……Z80という，今から思えばちっぽけな玩具のような8ビットCPUで，さまざまな夢を描いていた若者達がいたんだ。

彼らはたいてい無頼の輩（ハハハ）で，自分の腕だけを信じ，ほかの誰にもできないようなことをやってのけて，そしてニコニコと笑うのが大好きな奴らだった。プリフェッチやパイプラインもなく，ましてやキャッシュもなく，たったの4MHzで拍動するチップの上で，まるで一輪車に乗りながらお手玉をするようにレジスタを操っていたんだ。

なんでそんなことをしていたのかって？ うーん，……きっと彼らはそうしている時間に，ちょっぴり暴走のスリルというスパイスをかけて，長いけど短い幾つかの夜を過ごすことを，ただそれだけを好きだったんだろうね。

*

……それがやって来たのは，突然といえば突然だったな。でも，まあ，なんというか，やっぱり必然でもあったんだ。

もう今はどこにも使われてないけど，8086という16ビットCPUがあったんだ。それと，その8086とソフトでコンパチなんだけど，16ビットなのか8ビットなのかわかんない8088というCPUもあってね。それでね，日本では8086を使ったパソコンが，そして海の向こうのアメリカでは8088を使ったパソコンが登場したのさ。

ああ，大嫌いだったよ。虫酸が走ったね。特に“セグメント”という奴がね。あんなケチなからくりなんて最低だよ。ようするにエレガントじゃないんだ。小手先のゴマカシなんだよ。とりあえずは16ビットだったけど，でも，未来を見つめて作られたものじゃなかった。少なくとも“俺の未来”じゃなかったね。

でも，現実に5MHzの8086は，やっぱりそれなりの地力があったな。だけど，最大の成功の理由は，本当はCPUのパワーじゃなくて，あのマシンが漢字VRAMを持っていたことだろう。あれのおかげで，実用的な速度で漢字を処理することが可能になったんだ。その意味じゃ，あんまり馬鹿にはできないんだけど。

*

で，さ！ そして，その頃から，おかしなことをいうオトナ達が現れ始めたんだ。つまりね，「漢字は16ビットコードだから，16ビットCPUのマシンこそ日本語処理に適しているんですよ」なんてことを，したり顔で偉そうに講釈するオッサン達さ。

俺達はゲラゲラ笑ったよ，もちろん。そして，決して彼らの誤解を解いてやろうなんて思わなかった。だって，俺達は俺達で，オッサン達はオッサン達だったから。

そうだね，そして，きっとあの頃からだね。少しずつ，少しずつ，面白くないことや，くだらないことが増えていったのは。つまりさ，オッサン達が，パソコンは金になる，と考え始めた頃さ。

あの頃はデタラメだったな。本当にムチャクチャだった。俺もいろいろなデタラメを見たし，巻き込まれもしたよ。幸いなことに俺は結構カンがよかったからさ，運よく逃げ回ったけど。でね，逃げ回りながら，俺は少しずつ時代から取り残されていくZ80の上で，相変わらず一輪車に乗ってお手玉をしていたんだ。だから今でもお手玉には結構自信があるぜ，もちろん腕は多少錆びついちゃってるけどね。でも，まあ，普通に考えれば，アホだろうね。でもさ，そのアホも結果によるだろ。つまり，そうして待っている間に，やっとまともな16ビットパソコンが出てきたからね。

*

そりゃあそうさ！ 16Mバイトがリニアで，32ビット長のレジスタが16個なんだからね。だから，もう一輪車のお手玉なんかじゃなくなったわけさ。しかも，おまけにもうオッサン達のゴタゴタに関わらずに済むというオマケ付きだったんだ。だって，オッサン達は目先の金儲けに夢中で，だから，あっちの方向だけを見ていたわけさ。何ひとつわかってないくせに，“86系以外のCPUが成功するわけがない”なんて言い切ってさ。馬鹿だね。メジャーなほうに追随すりゃあ，一番デカイ奴にハジキ飛ばされるに決まってるじゃないか。そんなこともわかってなかったんだ，オッサン達は。

つまり，ああ，つまりだな，ようするに，秘密の空き地みたいなものさ，たとえばね。子供達が何かの拍子に，誰からも見つけられない秘密の遊び場を見つけたと思ってごらん。そこには何でもあって，そして，何をしても咎められない，そんな原っぱさ。チャンバラをしようが，プロレスごっこをしようが，爆竹を鳴らそうが，なんでもできるんだ。そんな場所を考えてごらん。全部をうまくはいえないけれど，でも，そんな感じが一番近い。上を見上げるとさ，真っ青に晴れ上がった空に，白い雲が流れているんだ。ただそこにいるだけで嬉しくなってしまうような場所。そんな場所さ。

だけど，青空はいつまでも青空というわけにはいかない。秘密の原っぱといえども，容赦なく時間が流れるからね。結局夕方には子供達も家に帰らなければならないだろ。そんなときに，ハハハ，あの8086の子孫の“バケモノ”が待ち構えてるわけだよ。

まったく狂ってるよ，まったく。マッドエンジニアだよ，あんなものを作り上げた奴らは，ほんと。リアルモードにプロテクトモード？ それに仮想8086モードやらなんやらかんやら。あいつら，いったい何考えてんだ？

……それは違うと思うな。俺にいわせりゃあ，あいつらはグルなんだよ，グル。新しいマシンを買わせるには，重いソフトを出して“標準”にすればいい。そうすれば，もっとクロックの速いCPUの載った，キャッシュも多くついたマシンに買い替えざるを得ないだろ？ グルなんだよ，グル。

*

たとえば，……買ってもらったばかりの5段変速の自転車。……明日から夏休みになる日の学校からの帰り道。もう，みんな忘れてしまっただろうけど，そんなようなことなんだな。おかしいかな？ まあ，俺はアホだから，そんなことにいつまでもこだわってるのかもしんないけどね。でもね，そうだね。う〜ん，新しいアドレッシングモードにワクワクしたり，ビットフィールド操作命令をどうやって使ってやろうかと考えたりとかね。ちょっと変かもしれないけど，そういうことなんだ。

*

結局これは終わりのないお話なんだ。だから，いつまでたっても序章なんだよ。そういうことなんだ。

夢？ そりゃまた陳腐な台詞だね。ま，そういう言い方もあるかもしれないけどね。

子供達はまだ知らない。大人達は忘れてしまった。

正規表現を利用したパターン照合

Murata Toshiyuki　村田 敏幸

ある条件を満たすすべての文字列を示すのに，正規表現という方法があります。今月は，これを文字列の照合/探索に利用してみましょう。このアルゴリズムにより，文字列のなかからあるパターンの複数の文字列を検索するルーチンを作成することができます。

前回は固定文字列の照合/探索アルゴリズムを取り上げた。今回はその延長線上の話題として，“パターン”に一致する複数の文字列をテキスト中から並行して探すことを考える。探索パターンの記述には，この種の目的に広く使われている“正規表現”を利用する。

正規表現

正規表現(regular expression：正則表現，または正則式などとも訳される)は，UNIX界ではお馴染みの文字列パターン記述方法だ。いくつかのメタ文字(特殊文字)を用いることで，文字/文字列の繰り返しや，論理和などを含んだパターンを比較的簡潔な形式で記述することができる。UNIXにはgrep/egrep, sed, awkなど，正規表現を利用した強力なツールが存在し，また，テキストエディタは当然のように正規表現による文字列探索をサポートしている。これらのプログラムはHuman68kにも移植されているから，日々活用しているユーザーも多いことだろう。

この正規表現，もともとは言語理論の分野において，ある種の“言語”の“文法”を記述する方法として生まれた。ここでいう文法とは，文字の並べ方のルールといった程度の意味で，言語とは，そのルールに従った文字列の集まりを指す。逆に，文法に従った文字列のことを，その言語の“文”である，といったりもする。正規表現を利用したパターン照合は，ある文字列が正規表現で表された言語の文かどうかを判定する操作の応用，というか，そのものだ。

そういうわけで，本来ならここで正規表現の定義を示すべきなのだと思う。だが，今回は正規表現を単なるパターン記述方法としてのみ扱い，理屈は抜きにして，記法を示すにとどめる。

まず，任意の文字cはその文字c自身とのみ一致する正規表現であり，また，空文字列は空文字列とのみ一致する正規表現だ。これらをつぎに示す演算子を使って組み合わせることで，より複雑な正規表現が得られる。

1)　論理和(選択)

r | sは2つの正規表現rとsのどちらかと一致する文字列と一致する。

2)　連結

r･sは2つの正規表現r, sそれぞれに一致する文字列が繋がった文字列と一致する。連結演算子「・」は通常省略する。

3)　閉包

r*は正規表現rに一致する文字列の0回以上の繰り返しを表す。たとえば，a*は，

　a，aa，aaa，aaaa，……

などのaだけからなる文字列，および，空文字列(=aの0回の繰り返し)と一致する。

ここで，演算子は，

　閉包　>　連結　>　論理和

の順に強く結びつく。算術式の場合と同様，この優先順位を変える目的でカッコを使うことができる。また，正規表現でも，代数的な交換法則，結合法則，分配法則などが成り立つ(図1)。この公理は，冗長な形で与えられた正規表現を最適化するのに応用できる。

現実のプログラムではこれ以外の拡張や省略形がサポートされていたり，また，表記が若干異なったりもするが，簡単のため，当面はいま示したピュアな正規表現に限って話を進める。実際にプログラムで使われる拡張正規表現については，最後にコラム(125ページ)にまとめておいたのでそちらを見てほしい。

図1　正規表現の代数的性質

```
r|s=s|r
(r|s)|t=r|(s|t)
(rs) t=r (st)
r (s|t) =rs|rt
(s|t) r=sr|tr
(r*)*=r*
```

有限オートマトン

天下りに示してしまうと，ある文字列が正規表現で表されたパターンと一致するかどうか調べる操作は，正規表現を認識する“有限オートマトン”を作成して，それをプログラムでシミュレートする，という方法で実現すると効率がよい[1]。

有限オートマトン(finite state automaton：有限状態機械)というのは，ごく単純な仕組みだけで構成された仮想的な“機械”だ。その機械はいくつかの(有限個の)“状態”をとることができ，最初は“初期状態”にある。ここから入力を順に読み込んでは，その入力に応じてある状態からある状態へと“遷移(推移)”する。この動作は入力が尽きるか，“最終状態”に達するまで続く。パターン照合を行う有限オートマトンの場合，入力として文字を1文字ずつ読み込み，最終状態に到達することができたら，入力された文字列がパターンに一致した，と判断する。逆にいうと，パターンに一致する文字列を読み込み終わったときにとるべき状態を最終状態と決める。その意味で，以下，最終状態の代わりに受理状態という言葉を使う。

有限オートマトンは，よく，流れ図の形で表される。図2に正規表現

(a | b)*aa

を認識する有限オートマトンの例を示す。図中，各状態は状態番号を○で囲み，とくに受理状態は◎で表してある。また，

$\xrightarrow{a}$

のようなラベルつきの辺は，“入力がラベルの文字(いまの場合a)と一致したら遷移する”ことを表す。

いま例に使った正規表現は，aとbだけからなり，末尾がaaであるような文字列，たとえば，

aa，aaa，baa，aaaa，abaa，baaa，bbaa，……

などと一致する。試しに図2の有限オートマトンに文字列aaを与えてみると，初期状態である状態1から，入力第1文字目のaによって状態2へ，さらに2文字目のaによって状態3＝受理状態に到達し，照合は成功する。また，入力としてbbaaを与えると，

$$1 \xrightarrow{b} 1 \xrightarrow{b} 1 \xrightarrow{a} 2 \xrightarrow{a} 3$$

というルートを通って受理される。

ここで，aaaは，

$$1 \xrightarrow{a} 2 \xrightarrow{a} 3 \xrightarrow{a} 3$$

を通って受理されることになるが，途中，aaまで読み込んだ状態もまた受理状態だ。この最初の受理状態で機械を停止させるかどうかは応用による。文字列の先頭部分がパターンと一致することがわかるだけで十分なら止めればいいし，文字列がパターンとぴったり一致するかどうかが知りたければ，入力が尽きるまで読んで，最後の状態が受理状態かどうかを調べなければならない。で，今回はテキスト中からの文字列探索に応用するわけだから，基本的には，いったん受理状態に達したら照合成功とみなして動作を打ち切ってよい。

ただし，一般に，正規表現は一致しうる最長の文字列と一致することになっている。たとえば，正規表現a*で表されるパターンを文字列aaaaと照合する場合，可能な選択肢は，

空文字列，a，aa，aaa，aaaa

とあるが，このうちで最も長いaaaaを採用する約束なのだ。これは，文字列置換に応用する際に，あいまいさをなくすためのルールだ[2]。このような最長一致を検出するためには，受理状態に達してもその時点での文字位置を記憶するだけにして，入力が尽

1）これ以外のより素朴な方法としては，正規表現の持つ再帰性に目をつけた再帰的なプログラムによる実現が考えられる。

2）このルールには“直後に続く正規表現の一致を妨げない範囲で”という条件がつく。a*aと文字列aaaaを照合する場合，a*をaaaaと一致させてしまうと，残りのaと一致する文字がないので照合が失敗してしまう。この場合，a*の一致範囲はaaaに制限しなければならないのだ。

図2　有限オートマトン

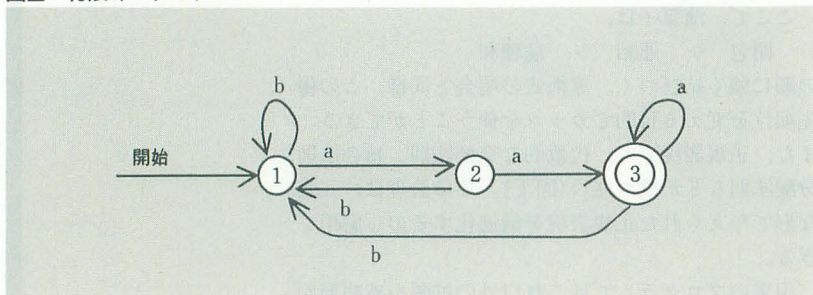
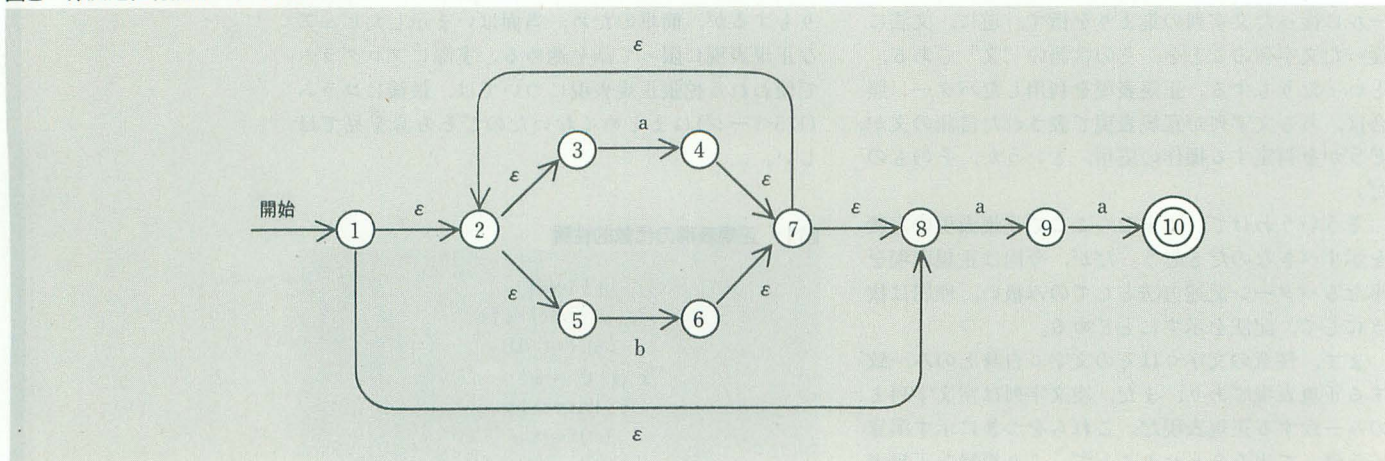



図3　非決定性有限オートマトン

きるか，不一致が検出されるまで動作を継続する。最後に受理状態に到達した時点の文字位置が最長一致文字列の末尾だ。

さて，図2のオートマトンでは，ある入力文字による，ある状態からの遷移先はただひとつしかない。このような有限オートマトンを“決定性(deterministic)有限オートマトン”(以下，DFAと略す)という。対して，入力文字に対応する複数の遷移先を許す有限オートマトンを“非決定性(non-deterministic)有限オートマトン”(NFAと略す)という。パターン照合に応用する際には，DFAを利用したほうが効率のよいパターン照合ルーチンが作れる。図2のDFAは最大でも入力文字列の文字数回の遷移で一致/不一致を判定することに注目してほしい。文字の比較回数だけを考えれば，固定文字列どうしの比較と変わらない手間だ。

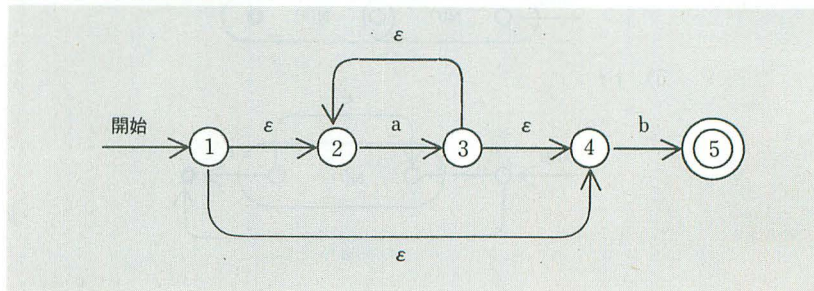
しかし，DFAはNFAに比べ，作成にやや複雑な処理を必要とする。メモリも比較的大量に使う。特に，最悪の場合，状態数が正規表現の文字/演算子の数と指数的に比例することが知られている。また，ある種の拡張正規表現と相性が悪く[3]，加えて，日本語(2バイト文字)に対応するのも難しい[4]。そこで今回は，性能は落ちるものの，作成が簡単で，メモリもあまり使わずに済み，日本語対応も十分可能な，NFAのほうを採用する。

図3に図2と同じ正規表現を認識するNFAの例を示す。図中，記号εは空文字列を表す。εによって遷移するということは，つぎの文字を読む前に遷移できることを意味する。感じとしては，入力の文字と文字の合間にある空文字列を読み込んで遷移するわけだ。ε遷移による枝分かれがあると，ある瞬間にとりうる状態が一意には決まらなくなるので，オートマトンは非決定性を持つことになる。理論上，複数の選択肢がある場合，NFAは受理状態に到達可能な遷移先を“推論”するとされる。推論する機械なんて嘘っぽいと思ったら，NFAは同時に複数の状態をとりうる，と考えてもよい。NFAは必要に応じて複数の状態に分裂して，以後，並列動作し，そのいずれかが受理状態に到達したら照合成功と判断する。受理状態には到達できないとわかった選択肢はふるい落とされていき，とりうる状態がなくなったら照合失敗だ。

いま示したNFAの2つの解釈は，そのままNFAをシミュレートする際の考え方となる。まず，“推論”を“試行錯誤”に置き換えて実現するアプローチがある。ε遷移による枝分かれがあったら，そのうちのひとつを選び，残りはスタックに押し込んでおく。最長一致を検出するためには，ε遷移によるループがもしあれば，そちらを先に処理する必要がある。また，入力を先読みする格好になるので，スタックにはNFAの状態だけではなく，対応する文字位置もセットにして記憶する。そして，選んだ選択肢をたどれるだけたどり，受理状態に到達しないとわかったらスタックから途中経過を復元して，そこからまた別のルートをたどる。スタックが空になったら照合失敗だ。

“並列”型は，現在とりうる状態を常にリストアップしておくことで実現できる。複数の状態を並列に睨みながら照合を進めるのだ。並列といっても，実際のプログラムでは同時にはひとつの状態しか処理できないので，スタックか，それに類した一時記憶場所はやはり必要になる。スタックを2本用意するのが一般的な方法だ。一方(仮にスタックA)にはいますぐチェックすべき状態を入れておく。ここから状態をひとつ取り出しては，遷移可能かどうか調べ，遷移できたらもう片方のスタック(B)に移す。ε遷移による枝分かれ先は，一時的にスタックAに押し戻し，つぎの文字を読まないうちに処理する。スタックAが空になったら2本のスタックの役割を交換し，つぎの文字を読んで処理を継続する。

2種類の実装にはそれぞれ一長一短があるのだが，あとで示すプログラムでは並列型を採用した。まず，並列型のほうが作業用メモリが少なくて済む。試行錯誤型では，入力文字列のすべての文字位置がスタックに積まれる可能性があるため，それを見越してスタックの大きさを決めなければならない。ところが，入力文字列の長さはプログラムには予想できないから，どんなに大きくスタックを確保してもスタックが溢れないという保証はない。結局，このぐらいでいいだろう，というところで妥協して，そのスタックの大きさで扱える範囲でテキストの長さを制限することになる(黙って飛ぶプログラムもあるけど)。並列型では，重複が起こらないよう注意しさえすれば，スタックには最大でもNFAの状態数分の情報しか積まれない。NFAの状態数は作成時にわかるから，その2倍(2本分)のメモリを確保しておけば十分だ。

また，試行錯誤型は，最悪の場合の実行速度が非常に遅い。この方法では，照合過程で不一致が検出されたらスタックから途中経過を復元して再試行する。このため，明らかに照合が失敗する場面でも，スタックが空になるまで無駄な再試行を続ける場合がある。たとえば，aだけからなるテキストとパターンa*bとの照合を試みる場合を考えよう。NFAは図4のようになる。最長一致を見つけるという前提だと，状態2→3→2のループを優先的に処理することになり，比較がテキスト末尾に達した時点で最初の不一致が見つかる。テキストにはbが含まれてい

3) 後方参照の実現は不可能だ(と思う)。実際，UNIXのegrep(DFAを利用するgrepの高速版)では後方参照がサポートされていない。GNUのgrep/egrepはDFAを利用するにもかかわらず，後方参照をサポートしているが，処理速度などから類推するに(ちゃんとソースを読んだわけではない)，正規表現が後方参照を含む場合はDFAを利用していない(少なくとも活かしていない)ように見える。

4) DFAを構成する過程で，“入力として有効なすべての文字”について処理を反復する箇所があるため。日本語は文字の種類が多すぎるのだ。

図4 a*bを認識するNFA

ないから，これ以上照合を続けても無駄なのだが，この時点でスタックにはすべてのaの文字位置が積まれており，これを処理し終わるまでプログラムは止まらない。スタックからひとつ途中経過を復元するたびにテキスト末尾まで比較を繰り返し，その過程でもスタックに途中経過を積むというひどい事態になる。a*a*bとかになるとさらに処理時間は膨れ上がる。並列型では，テキストの各文字は最大でもNFAの状態数回しかチェックされず，あと戻りもないから，文字の比較回数は入力の文字数×NFAの状態数に収まる。少なくとも，処理時間が爆発的に膨れ上がるようなことはない[5)]。

とはいえ，並列型にもそれなりの問題はある。試行錯誤型では，可能な選択肢をひとつだけ追い，残りはあと回しにするから，運がよければ無駄な文字比較なしの最短時間で照合を検出することができる。並列型ではすべての可能性を並行して扱うがために，試行錯誤型ではしなくても済んだ文字比較を行うことがありうる。さらに，並列型では，一般のプログラムで採用されている拡張正規表現の一部がサポートしにくい。この点については，あとでプログラム例を示すときにもう一度振り返る。

5） やりようによっては，試行錯誤型でもスタックに無駄な途中経過を記憶しない（しにくい）ようにすることは可能だ。実際，GNUの正規表現ライブラリにはそのような工夫が見られるようだ。

正規表現からNFAへ

正規表現を認識するNFAは，もとの正規表現から機械的に作成することができる。そのアルゴリズムでは，正規表現を構文上のパーツに分解し，個々の正規表現を認識する小さなNFAを作り，それを組み合わせて最終的なNFAを作成する。基本は図5の5パターンだ。なお，図5では正規表現rを認識するNFAをN(r)で表している。

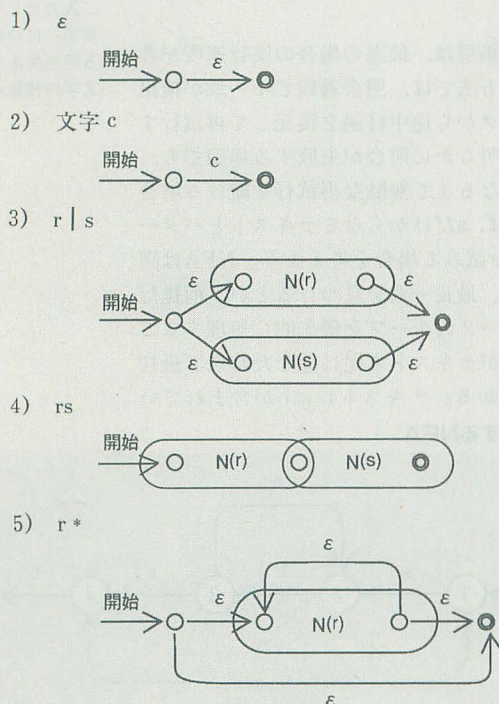



図5　正規表現からNFAへの変換

1） ε

2） 文字 c

3） r｜s

4） rs

5） r*

1)，2)，3)については，見たままだから説明するまでもないだろう。3)では論理和を実現するために，N(r)とN(s)をε遷移により並列に繋いだNFAを作成している。新たに設けた開始状態から受理状態へ抜けるには，N(r)かN(s)のどちらかを通過することになり，この正規表現はr｜sを正しく認識する。

4)は，連結演算を実現するには，N(r)の受理状態とN(s)の開始状態を重ねて直列に繋げばよいことを示している。

5)の閉包r*を認識するNFAはやや複雑だが，順を追って考えてみれば納得してもらえると思う。まず，0回の繰り返し（＝空文字列）は，初期状態から受理状態へのε遷移によるバイパスを通って受けつけられる。1回の繰り返しの場合は，N(r)を抜けて受理状態に達する。2回以上の繰り返しの場合は，N(r)の受理状態から初期状態へのε遷移によるループを必要なだけ回ってから受理状態へと抜ける。

どのパターンももとの正規表現のニュアンスを残しているので，動作は容易に把握できるだろう。また，これらを組み合わせれば，より複雑な正規表現を認識するNFAが出来上がることも，直感的に理解できるに違いない。むしろやっかいなのは，正規表現を構文単位に分解して適切に組み合わせる操作だ。この操作はいわゆる構文解析であり，コンパイラとかインタプリタなどの言語処理系の領域になる。確かに，正規表現を内部形式のNFAに変換する操作は，一種のコンパイルに違いない。大それたことに，いま僕たちは，優先順位つき演算子やカッコを含む式のコンパイラを作ろうとしているのだった。もっとも，正規表現の再帰性に着目すると，小難しい理屈は知らなくてもプログラムは書ける。

カッコはないものとして，演算子の優先順位を実現するところまでを示そう。まず，正規表現とは，論理和を含まない部分正規表現が，論理和演算子「｜」を挟んで1個以上並んだもの，と考える。さらに，その部分正規表現は，論理和も連結を含まない，もうひと回り小さな単位の部分部分正規表現が1個以上連結されたもの，と置く。その部分^{2}正規表現となりうるのは，任意の1文字か，その閉包だ。ここで，まとめた単位ごとにサブルーチンを用意する。

1） 正規表現をNFAに変換するサブルーチン
2） 部分正規表現をNFAに変換するサブルーチン
3） 部分^{2}正規表現をNFAに変換するサブルーチン

3)のサブルーチンは単独の文字か，その閉包のNFAを作成するだけなので，比較的簡単に実現できる。で，3)を下請けに利用すると，2)のサブルーチンの実現も容易だ。3)に作らせたNFAを組み合わせるだけの処理を行えばよい。同様に，2)を利用すれば，1)のサブルーチンもすぐにできる。これで，演算子の優先順位つき構文解析が実現される。

残るはカッコだが，カッコでくくられた正規表現は，ちょうど1個の文字と同列の扱いになることを

考えると，3)の中で処理するのが自然だ。また，カッコの中身は独立した正規表現だから，1)を使えば処理できる。つまり，3)から1)を呼び出すという間接再帰を利用することで，カッコも正しく処理できるようになる[6]。

あと，考えておかなければならないのは，NFAの内部表現だ。NFAの流れ図としての性格を考えると，各状態は，遷移する条件となる文字と遷移先を指すポインタ(状態番号でもいいし，アドレスでもいい)の組で表現できる。だが，先に示したアルゴリズムで作成されるNFAの各状態は，多くても2本の辺しか持たず，しかも，ありうる可能性は，

1)　文字をラベルに持つ辺が1本だけある
2)　εをラベルに持つ辺が1本だけある
3)　εをラベルに持つ辺が2本ある

という3通りしかない。そこで，今回は配列にべたっと詰めて格納することにした。1)の場合，配列には遷移するための条件のみを格納し，遷移先は直後のメモリにあると決めつける。2)は論理和演算子で枝分かれしたNFAを束ねるのに使うので，遷移先，というか，ジャンプ先へのポインタで表す。3)については，片方の遷移先はポインタで表し，もう片方は配列の位置関係を利用することでポインタを省略する。実際には，これら3通り，および，そのほかの拡張表記を区別するために，内部的な識別番号をつけ加える。

プログラム例

では，正規表現で表されたパターンをテキストから探すプログラムの実装例を示すとしよう。リスト1～4だ。リスト1が定数/構造体定義，リスト2が文字種判別用の簡単なマクロ定義，リスト3が正規表現からＮＦＡへの変換部，リスト4が探索/照合処理本体となっている。動作試験にはリスト5を利用してほしい。リスト3，4は，下請けに先月作成したＢＭ法による文字列照合ルーチン(1993年3月号60ページのリスト7)と，文字種判別テーブル(同61ページのリスト9)を使っているので，一緒にまとめてリンクすること。出来上がった実行ファイルは，ファイル名を与えて起動すると正規表現の入力待ちになる。適当に入力すると，起動時に指定したファイル(の先頭16Kバイトの範囲)から一致する行を探し，一致部分を色を変えて表示する。

今回のプログラムでは，別コラムに示した拡張を含む正規表現のメタ文字のうち，「¥1」～「¥9」による後方参照と，「¥{m，n¥}」による固定回数の繰り返しを除く，すべてをサポートしている。コラムでも触れたように，本来，後方参照は正規表現にはない概念なので，有限オートマトンとは相性が悪い[7]。固定回数の繰り返しは，繰り返しを展開する方法で実現できないこともないが，ちょっと冗長すぎるので，やめた[8]。その代わりといってはなんだが，日本語にも正しく対応しており，2バイト文字もちゃんと1文字として扱うようになっている。

プログラムは全部合わせると結構な分量だ。全行を解説するのはちょっと無理そうなので，ポイントだけ拾っていこう。

●リスト1

まず，7～10行でコンパイル後のパターンの内部構造を定義している。先に触れたような手順でワード配列の形で作成したNFAに1ワードのモードフラグをつけた形だ。ここでいうモードフラグは，特定条件でのみ働く省力化ロジックが有効に使えるかどうかを，各1ビットのフラグで表したものだ。いまのところ，ごく簡単な3通りをサポートしてある(15～21行)。

26～46行は正規表現構文要素の内部コード定義だ。　.offsetにより，PascalやＣ風の列挙定数，つまり，通し番号のついた記号定数群として定義している。ここでは，.ds.bではなく，.ds.wを使っているのがミソだ。ふつうに，0，1，2，……，と番号を振らずに，0，2，4，……の偶数のみを使っている。構文要素の内部コードがジャンプテーブルのインデックスとして頻繁に使われることを考慮して，最初から2倍してあるのだった。

先に触れたように，構文要素によっては，この内部コードの直後に付随情報がつく場合がある。たと

6)　ここで示した構文解析ルーチンは，いわゆる再帰下降パーサの一例だ。この考え方は，そのまま通常の算術式のコンパイル/評価ルーチンになどに応用できる。

7)　NFAを試行錯誤型で実現する場合は，後方参照にも問題なく対応できる。

8)　試行錯誤型ならループカウンタを導入するような方法もとれる。

リスト1　RE.H

```
 1: *           正規表現によるハターン照合ルーチン用
 2: *                       定数， 構造体定義
 3:
 4: *
 5: *           コンパイル'ed パターンの構造
 6: *
 7:                         .offset 0
 8: *
 9: reMODE:                 .ds.w   1           *モード
10: reNFA:                  *.ds.b  ??          *< 32768バイト
11:
12: *
13: *           モード
14: *
15: reFASTRETN_BIT  equ     0           *照合が得られたらすぐ戻る
16: reBOLSCAN_BIT   equ     7           *行頭からとのみ照合する
17: reBMSCAN_BIT    equ     15          *頭部をBM法で探す
18: *
19: reFASTRETN      equ     1.shl.reFASTRETN_BIT
20: reBOLSCAN       equ     1.shl.reBOLSCAN_BIT
21: reBMSCAN        equ     1.shl.reBMSCAN_BIT
22:
23: *
24: *           正規表現内部コード
25: *
26:                         .offset 0
27: *
28: reTERM:                 .ds.w   1           *終端 または ]
29: reLITERAL:              .ds.w   1           *通常の文字
30: reDOT:                  .ds.w   1           * .   ,
31: reCLASS:                .ds.w   1           * [
32: reNCLASS:               .ds.w   1           * [
33: reCARET:                .ds.w   1           * ^
34: reDOLLER:               .ds.w   1           * $
35: reLBRACKET:             .ds.w   1           * ¥<
36: reRBRACKET:             .ds.w   1           * ¥>
37: reLPAREN:               .ds.w   1           * ¥(
38: reRPAREN:               .ds.w   1           * ¥)
39: reREFER:                .ds.w   1           * ¥1 ~ ¥9 (未対応)
40: reASTERISK:             .ds.w   1           * *
41: rePLUS:                 .ds.w   1           * +
42: reQMARK:                .ds.w   1           * ?
43: reVBAR:                 .ds.w   1           * ¥|
44: reMAIN:                 .ds.w   1           *ε遷移による枝分かれ(本流)
45: reBRANCH:               .ds.w   1           *ε遷移による枝分かれ(支流)
46: reBIND:                 .ds.w   1           *ε遷移による合流
47:
48: *
49: *           照合成功位置の記録用ワーク構造
50: *
51:                         .offset 0
52: *
53: reHEAD:                 .ds.l   1           *照合成功位置先頭
54: reTAIL:                  ds.l   1           *照合成功位置末尾+1
55: SIZEofREREG:
56:                         .text
```

えば，通常の文字を表すreLITERALの場合なら，直後にその文字コードが続く。このあたりの細部の構造は図6にまとめておいた(図にないものは付随情報を持たない)。「[]」内に列記した文字のいずれか1文字と一致する構文である文字クラスの内部表現については，やや説明を要するだろう。今回は，複数文字との比較が簡単に行えるよう，「[]」内にどんな文字が含まれるかを，256ビットのテーブルで表すようにしてある。文字コードに対応するビットを調べれば，その文字が文字クラスに含まれているかどうかがすぐわかるわけだ。文字コードとテーブル上のビット位置は，文字コードの小さい順に，低位アドレスから上位アドレス，また，各バイト中では下位ビットから上位ビットの順に対応づける。この方法では対応しきれない2バイト文字はテーブルの直後に並べておき，線形探索する。この2バイト文字リストは「[あ-ん]」のような範囲指定を考慮して，常に下限と上限の組で表すことにした。また，テキスト中に2バイト文字が出てくるたびに文字リストを探すのでは遅いので，事前に大まかな判定ができるよう，ビットテーブルに上位バイトについての情報を登録するようにしてある。

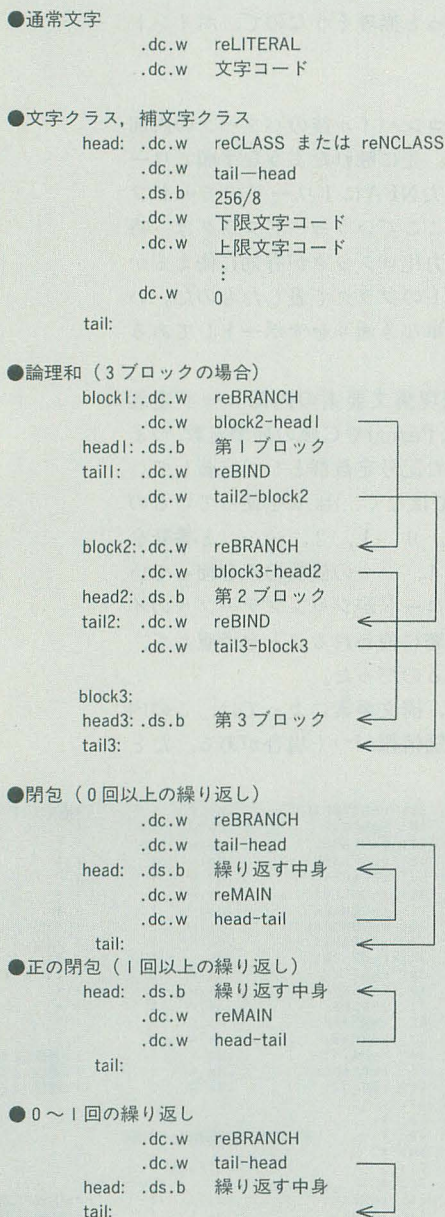



図6　今回採用した内部コード

あと，ε遷移を表す内部コードが3通り用意してある点をチェックしておいてほしい。今回のプログラムでは，ε遷移をただひとつ持つ状態と，2つのε遷移によって枝分かれする状態を区別できれば十分なのだが，枝分かれをさらに2種類に分類した。試行錯誤型などへの応用を考慮し，最長一致を得るためには2つの遷移先のどちらを先にたどったらよいか，を内部コードで区別できるようになっている。リスト3のサブルーチンre_compは2種類の枝分かれを区別したNFAを作成するから，試行錯誤型の照合ルーチンにもそのまま流用できる。

51〜55行はパターンに一致した文字列の位置を格納する構造体の定義だ。照合部本体は見つけた文字列をこの形式でモジュール内部の静的なワークに格納し，その先頭アドレスをa0に返す。

●リスト2

先月のリスト9のテーブルを利用した文字種判別マクロを定義している。これらのマクロを使うときには，文字種判別テーブルの先頭アドレスを適当なアドレスレジスタに入れ，そのレジスタを

```
	__CTYPETABLE__reg a0
```

のように定義しておく。そのうえで，たとえば，d0.w(上位バイトは0であること)が半角英小文字かどうかで処理を振り分けたければ，

```
	ISLOWER	d0
	bne	条件成立時の処理
	条件不成立時の処理
```

のように書く。マクロBZENはちょっと変なマクロで，

```
	BZEN	d0,label
```

のように使い，レジスタに格納した文字コードが2バイト文字の第1バイトだったらlabelに分岐する。同様に，BHANは1バイト文字だったら分岐する。

●リスト3

正規表現→内部形式NFAのコンパイル部だ。サブルーチンre_compには引数として，正規表現とNFAの格納領域(とその末尾)を与える。エラーなくコンパイルできたら，その先頭アドレスをa0に，バイト数をd0に返す。構文上のエラーなどが見つかった場合は，a0にエラーメッセージの先頭アドレスを入れ，ccrのNビットを立てて戻る。

やや大きなプログラムだが，比較的きれいにまとまっていると思うので，ブロック単位で読んでみてほしい。先ほど触れた3段階の構文解析を実現しているのが，174〜545行だ。174〜220行，222〜234行，236〜545行の順に処理単位が小さくなっていく。最

小の1構文単位を処理する236〜545行では，jump tableで処理を振り分けて，各構文要素に個別対応している。文字クラスの処理(342〜486行)中の，ビットテーブルを扱う下請けサブルーチンbitfield (429〜486行)がやや難解にみえるかもしれない。ここでは，水平型VRAMに水平線分を引く要領で，d0で指定されたビット位置からd1で指定されたビット位置までを1で埋めている。d0, d1に適当な値を入れたときの動作を卓上で追ってみるとよいだろう。

閉包を含む繰り返し構文を扱う488〜545行では，重複した繰り返し指定を最適化するようにしてみた。コメントに示したような変形が常に成り立つことを確認しておいてほしい。

547〜668行は字句解析部で，正規表現のつぎの文字を内部コードに変換してd0に，付随情報をd1に返す。メタ文字については，「¥」+1文字のものと，「¥」がつかないものをそれぞれテーブル(584〜591行，617〜622行)にしてあり，簡単に変更/追加できるようになっている。たとえば，リスト3では「¥｜」で論理和を，「｜」で文字「｜」自身を表すものとしている，この意味を逆にしたければ，619行を584〜591行のテーブル中に移動すればよい。なんなら，実行時にテーブルを組み替えることも可能だろう。

NFAが出来上がったら，125行にくる。ここでは，のちの照合部の手間を省くための細工をしている。ひとつは，正規表現先頭に固定文字列があるかどうかの検査だ。もしあれば，その文字列をBM法で探してから照合することで，処理時間の短縮が望める。リスト3では，125〜143行でNFAを初期状態からたどり，文字を読まずに，かつ，ε遷移による枝分かれもなしに到達できる最初の状態が，通常文字かどうかを調べている[9)]。そうだったら，145〜158行でスタック上にふつうの文字列として再構成し，bm_compでBM法の前処理を行っておく。

もうひとつの小細工は，行頭とのみ一致する拡張正規表現のメタ文字「^」がパターン先頭にあるかどうかのチェックだ。この検査は固定文字列を探す処理の過程で行える。もしあったら，フラグを立てておき(142行)，照合はテキスト各行の先頭部分とのみ行えばよいことを照合部に伝えている。

この種の省力化案はほかにも考えられるから，組み込んでみるのもいいだろう。

●リスト4

リスト4のサブルーチンre_execが探索/照合部本体だ。先に示した“並列型”でNFAをシミュレートする。ただ，目新しさを狙って，2本のスタックを使う代わりにデクと呼ばれるデータ構造を使ってみた。デク(deque)はDouble Ended QUEueの略で，データの追加/取り出しを両端から行うことのできるキューだ(図7)。リスト4では，このデクの前半部にすぐに処理しなければならない状態を格納し，区切りのダミーデータ(0)を挟んで，後半部にあとで処理する状態を格納する。先ほどの説明の2本のスタックの底を合わせて繋いだと思えばよい。デク先頭側から状態を取り出そうとしたときにダミーデータが読み出されたら，それをデク末尾に移動する。これが，ちょうどスタックの交換に相当する操作になる。

サブルーチンre_execには，58〜63行のような引数を与える。テキスト末尾は00_Hで表されるものとするが，BM法を利用する都合で，これとはべつにテキスト末尾のアドレスも指定することになっている。テキスト中にはCR+LF，または，単独のLFで区切って複数行を並べてあってもよい。単独のLFも行末として認識するようにしたのは，Cプログラムから呼び出すことを想定したためだ。ただし，リスト4はCの関数呼び出し規約に従っていないから，Cから呼び出すためには，サブルーチン名や引数/戻り値の受け渡し方法を微調整する必要がある。

最後の引数である“直前の文字”は，1行中からパターンに一致するすべての文字列を探す場合に，拡張正規表現「^」や「¥<」「¥>」が誤動作しないように設けた。通常は，$000A_H$(LFコード)にしておけばよい。あとから強引に加えた引数なので，仕様はちょっと変かもしれない。テキスト先頭以外に，照合開始位置を引数で渡すようにしたほうが自然だ

9) 正規表現レベルで行わず，NFAレベルで行うのは，¥(abc¥)を見落としたり，abc¥｜defを誤認したりしないようにするためだ。

リスト2 MYCTYPE.H

```
 1: *            文字種判別マクロ
 2:
 3: CDIGIT_BIT       equ     0
 4: CHEX_BIT         equ     1
 5: CALPHA_BIT       equ     2
 6: CUPPER_BIT       equ     3
 7: CLOWER_BIT       equ     4
 8: *                equ     5
 9: CHAN2_BIT        equ     6
10: CKANJI_BIT       equ     7
11: *
12: CDIGIT           equ     1.shl.CDIGIT_BIT
13: CHEX             equ     1.shl.CHEX_BIT
14: CALPHA           equ     1.shl.CALPHA_BIT
15: CUPPER           equ     1.shl.CUPPER_BIT
16: CLOWER           equ     1.shl.CLOWER_BIT
17: CHAN2            equ     1.shl.CHAN2_BIT
18: CKANJI           equ     1.shl.CKANJI_BIT
19: *
20: ISDIGIT macro   dreg
21:         btst.b  #CDIGIT_BIT,0(__CTYPETABLE__,dreg)
22:         .endm
23: *
24: ISHEX   macro   dreg
25:         btst.b  #CHEX_BIT,0(__CTYPETABLE__,dreg)
26:         .endm
27: *
28: ISALPHA macro   dreg
29:         btst.b  #CALPHA_BIT,0(__CTYPETABLE__,dreg)
30:         .endm
31: *
32: ISUPPER macro   dreg
33:         btst.b  #CUPPER_BIT,0(__CTYPETABLE__,dreg)
34:         .endm
35: *
36: ISLOWER macro   dreg
37:         btst.b  #CLOWER_BIT,0(__CTYPETABLE__,dreg)
38:         .endm
39: *
40: ISHAN2  macro   dreg
41:         btst.b  #CHAN2_BIT,0(__CTYPETABLE__,dreg)
42:         .endm
43: *
44: ISKANJI macro   dreg
45:         btst.b  #CKANJI_BIT,0(__CTYPETABLE__,dreg)
46:         .endm
47: *
48: CTEST   macro   typereg,dreg
49:         and.b   0(__CTYPETABLE__,dreg),typereg
50:         .endm
51: *
52: BHAN    macro   dreg,adr
53:         tst.b   0(__CTYPETABLE__,dreg)
54:         bpl     adr
55:         .endm
56: *
57: BZEN    macro   dreg,adr
58:         tst.b   0(__CTYPETABLE__,dreg)
59:         bmi     adr
60:         .endm
```

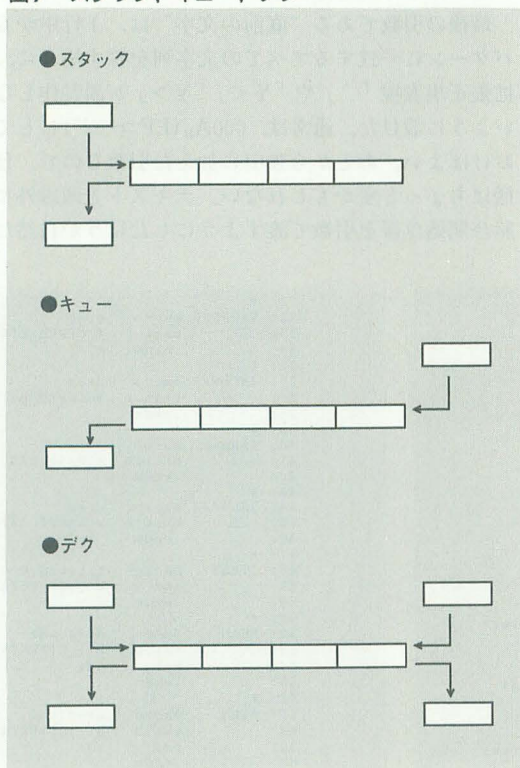
ったような気がする。

変な仕様といえば，re_execには事実上の隠れ引数がある。引数として渡すコンパイル済みパターンのモードフラグの第0ビットを立てておくと，一致する文字列を見つけたらすぐに戻るようになる。このビットが0なら一致する最長の文字列を探す。

さて，リスト4の中心は，サブルーチンeclosure(234〜339行)と，transit(341〜438行)だ。eclosureは，つぎの文字を読む前に遷移できるすべての状態をピックアップする。transitは実際に文字を読み込んでから，ある状態からその文字によって遷移可能かどうか調べる。

eclosureでは同じ状態を何度もチェックしたり，あるいは，重複してデクに格納したりしないよう，通過した状態のすべてに印をつけている(240行)。re_compが作成したNFAでは構文要素の内部コードの格納に2バイトを割いているが，実際には上位バイトは空いていることから，ここをフラグに使っている。立てたフラグは1文字の処理が終わった段階で，440〜474行のサブルーチンclrflagでクリアする。

ここで，行末を表す「$」など，文字そのものではなく文字の位置を表すメタ文字の扱いに注意したい。「$」はつぎの文字を読んでみないと一致するかどうかが判定できない。といって，「$」は行終端コードそのものと一致するのではなく，その直前の空文字列と一致するので，実際につぎの文字を読んでしまうのはまずい。この問題をリスト4では“先読み”により解決した。位置を表すメタ文字の処理はeclosure側で行い，こっそりつぎの文字を盗み見て，遷移できるかどうかを判定している。

図7 スタック，キュー，デク

10) 本文ではとくに触れなかったが，コンパイラの字句解析は正規表現の主要な応用だ。

eclosureとtransitを使い，154〜194行でテキスト中の部分文字列とパターンとの照合を行う。あらかじめ，文字を読まないうちに初期状態から遷移できるすべての状態をeclosureで探しておく(169〜173行)。それから，1文字読み込む(176〜194行)。ここでは2バイト文字も1文字として扱う。読み込んだ文字が行の終端コードだった場合は，このテキスト行とはこれ以上照合を続けてもしかたないのですぐにループを抜ける。また，00_Hだったらテキスト末尾なのでやはりループを抜ける。どちらでもなければ，現在とりうる各状態についてtransitを呼び出し，遷移できるかどうかを調べる(157〜161行)。その結果，とりうる状態がなくなったらループを抜ける。まだあれば，154行に戻り，処理を繰り返す。

ループを抜けたあとで，一度でも受理状態に到達していたかどうかを調べる。到達していたら，そのときの文字ポインタの位置を最長一致文字列の末尾とし，照合開始位置と組にしてワークに格納したうえで，その先頭アドレスをa0にもち，照合成功で戻る。このとき一緒に一致した文字列の長さをd0に入れて返すようにしてみた。

まだ一度も受理状態に到達していなかったら，照合開始位置を1文字ずらして(125〜131行)，再照合を試みる。ここで，例の小細工が有効な場合は，BM法を使ったり(141〜151行)，行末まで飛ばしたり(114〜122行)して，一致が得られる可能性のあるつぎの文字位置までポインタを大きく進める。

さらに知りたい人のための参考文献

正規表現によるパターン照合についてより詳しく知りたい人は，Aho，Sethi，Ullmanの『Compilers』を読むことを勧める。翻訳は『コンパイラ』のタイトルでサイエンス社より2分冊で出ており，その1冊目に多くの関連アルゴリズムが示されている[10]。また，前回も参考文献に挙げたSedgewickの『Algorithms』にもPascalによる実現例がある。さらに，古典的名書であるKernighanとPlaugerの『Software Tools』(翻訳は共立出版より『ソフトウェア作法』)には，正規表現そのものではないが，それによく似たパターン照合のより素朴な例が取り上げられている。理論面の裏づけがほしければ，“オートマトン”とか“言語理論”とかの題がついた教科書にもあたってみてほしい。

パソコン誌では，『インターフェース』の1991年4〜7月号で正規表現とオートマトンが取り上げられていた。へたな教科書よりは簡潔にまとまっていて読みやすいかもしれない。また，うろ覚えだが，同時期の『スーパーアスキー』にも同様の記事があったように思う。もっと昔になると，『アスキー』の1988年7〜10月号にX68000用のアセンブリ言語による正規表現認識ルーチンのコード例と解説がある(とくによいプログラムというわけではない)。

リスト3 RECOMP.S

```
  1: *          正規表現によるパターン照合
  2: *          (パターンのプリコンパイル部)
  3:
  4:                    .include        re.h
  5:                    .include        myctype.h
  6:                    .include        const.h
  7:
  8:                    .xdef   re_comp
  9:                    .xref   bm_comp
 10:                    .xref   ctypetable
 11: *
 12: __CTYPETABLE__     reg     a4
 13:
 14: *
 15: *          コンパイル結果を登録する
 16: *
 17: PUT1               macro   data1
 18:                    subq.w  #2,d6
 19:                    bcs     toolong
 20:                    move.w  data1,(a1)+
 21:                    .endm
 22: *
 23: PUT2               macro   data1,data2
 24:                    subq.w  #4,d6
 25:                    bcs     toolong
 26:                    move.w  data1,(a1)+
 27:                    move.w  data2,(a1)+
 28:                    .endm
 29:
 30: *
 31: *          a1直前のd0.wバイトをずらしてnbytesの隙間を作る
 32: *
 33: *                    d0  a1
 34: *                    ┌──┐↓
 35: *                  □□□□□□_
 36: *
 37: *                    ↓
 38: *
 39: *                 nbytes d0   a1
 40: *                   ┌──┐ ┌──┐↓
 41: *                  □□_____□□□□_
 42: *
 43: INSERT             macro   nbytes
 44:                    local   loop,done
 45:
 46:                    subq.w  #nbytes,d6
 47:                    bcs     toolong
 48:
 49:                    movea.l a1,a5
 50:                    addq.l  #nbytes,a1
 51:                    movea.l a1,a6
 52:
 53:                    move.w  d0,d1
 54:                    lsr.w   #1,d1
 55:                    subq.w  #1,d1
 56: loop:              move.w  -(a5),-(a6)
 57:                    dbra    d1,loop
 58: done:
 59:                    .endm
 60:
 61: *
 62: *          先読みした正規表現1単位を押し戻す
 63: *
 64: PBACK              macro
 65:                    move.w  d1,d3
 66:                    swap.w  d3
 67:                    move.w  d0,d3
 68:                    .endm
 69:
 70: *
 71: *          re_compの引数構造
 72: *
 73:                    .offset 4
 74: *
 75: pat:               .ds.l   1                  *パターン
 76: work:              .ds.l   1                  *変換結果格納領域
 77: workend:           .ds.l   1                  *  同       末尾
 78: *
 79:                    .text
 80:                    .even
 81: *
 82: re_comp:
 83: SAVREGS =          d1-d7/a1-a6
 84: SAVSIZ  =          (7+6)*4
 85:                    movem.l SAVREGS,-(sp)
 86:                    move.l  sp,d7
 87:
 88:                    movem.l pat+SAVSIZ(sp),a0-a2
 89:                    move.l  a2,d6
 90:                    sub.l   a1,d6              *d6 = ワーク残りバイト数
 91:                    bcs     nomem
 92:
 93:                    moveq.l #0,d3
 94:                    moveq.l #0,d4
 95:                    moveq.l #0,d5
 96:
 97:                    PUT1    d5                 *モード(仮)
 98:
 99:                    cmpi.l  #$0000_8000,d6     *NFAの最大バイト数を
100:                    bcs     noshrink           *  15ビットに制限
101:                    move.w  #$7fff,d6          *
102:
103: noshrink:          lea.l   ctypetable,a4
104:
105: *          d0~d2   作業用
106: *          d3      先読みしたトークンの押し戻し用
107: *          d4      閉じた「¥)」のフラグ
108: *          d5      「¥(」のカウンタ
109: *          d6      ワーク残りバイト数
110: *          d7      エラー時のsp
111: *          a0      ソース側ポインタ
112: *          a1      デスティネーション側ポインタ
113: *          a2      論理和の区切り
114: *          a3      繰り返しの単位先頭
115: *          a4      文字種判別テーブル
116: *          a5~a6   作業用
117:
118:                    bsr     orcomp             *コンパイル
119:
120:                    tst.w   d0                 *d0 = reTERM | reRPAREN
121:                    bne     unmatchedparen     *右カッコが多い
122:
123:                    PUT1    d0                 *reTERM
124:
125:                    movea.l work+SAVSIZ(sp),a0 *a0 = コンパイル済み
126:                                               *         パターン先頭
127:                    moveq.l #0,d4              *d4 = モード
128:                    lea.l   reNFA(a0),a2       *a2 = NFA先頭
129: headloop:          move.w  (a2)+,d1           *NFA先頭から
130:                    beq     setscanmode        *  入力を読まないうちに
131:                    cmpi.w  #reLITERAL,d1      *  枝分かれせず到達できる
132:                    beq     foundhead          *  最初の状態を探す
133:                    cmpi.w  #reCARET,d1        *
134:                    beq     setbolmode         *
135:                    bcs     setscanmode        *
136:                    cmpi.w  #reLPAREN,d1       *
137:                    bcs     headloop           *
138:                    cmpi.w  #reREFER,d1        *
139:                    bcc     setscanmode        *
140:                    addq.w  #2,a2              *
141:                    bra     headloop           *
142: setbolmode:        ori.w   #reBOLSCAN,d4      *「^」を見つけた
143:                    bra     headloop           *
144:
145: foundhead:         lea.l   -256(sp),sp        *それが reLITERAL なら
146:                    movea.l sp,a3              *  正規表現先頭に
147:                                               *  固定文字列がある
148:
149:                    move.w  #255-1,d3          *その文字列をスタック上に
150: makestrloop:       move.w  (a2)+,d2           *  再構成(最大255文字)
151:                    bpl     makestrhan         *
152:                    subq.w  #1,d3              *
153:                    bls     makestrbreak       *
154:                    move.b  -2(a2),(a3)+       *
155: makestrhan:        move.b  d2,(a3)+           *
156:                    cmp.w   (a2)+,d1           *
157:                    dbne    d3,makestrloop     *
158: makestrbreak:      sf.b    (a3)               *
159:
160:                    pea.l   (sp)               *BM法の前処理を行う
161:                    jsr     bm_comp            *
162:                    lea.l   256+4(sp),sp       *
163:
164:                    ori.w   #reBMSCAN,d4       *BM法を使用するモード
165:
166: setscanmode:       move.w  d4,(a0)            *モードを格納
167:
168:                    move.l  a1,d0              *d0 = NFA末尾
169:                    sub.l   a0,d0              *d0 = ワーク使用量
170:                                               *N=0
171: compretn:          movem.l (sp)+,SAVREGS
172:                    rts
173:
174: *
175: *          r1|r2|r3| ... のコンパイル
176: *
177: orcomp:
178:                    movea.l a1,a2
179:                    bsr     conccomp           *論理和を含まない
180:                                               *  1ブロックをコンパイル
181:                    cmpi.w  #reVBAR,d0         *「|」で終わっていた?
182:                    bne     orretn
183:
184:                    PUT2    #reBIND,d0         *合流用のジャンプを追加
185:                                               *d0はこの時点ではダミー
186:
187:                    move.l  a1,d0              *              ┌・・・・→
188:                    sub.l   a2,d0              *   → ・ →r1 ─┘ next
189:                    INSERT  4                  *      |         ↑
190:                    move.w  #reBRANCH,(a2)+    *      └─────┘
191:                    move.w  d0,(a2)+           *
192:
193: orloop:            movea.l a1,a2
194:                    bsr     conccomp           *論理和を含まない
195:                                               *  1ブロックをコンパイル
196:                    cmpi.w  #reVBAR,d0         *「|」で終わっていた?
197:                    bne     orbreak
198:
199:                    move.l  a1,d0              *未解決だった
200:                    sub.l   a2,d0              *  reBINDの飛び先を格納
201:                    addq.w  #4,d0              *
202:                    move.w  d0,-2(a2)          *
203:
204:                    PUT2    #reBIND,d0         *合流用のジャンプを追加
205:                                               *d0はこの時点ではダミー
206:
207:                               *                ┌────────┬・・・・→
208:                               *   → ・ →r1 ─┘ ┌→ ・ →r2 ─┘ next
209:                               *      |          |  |         ↑
210:                               *      └─────┘  └─────┘
211:                               *
212:                    INSERT  4
213:                    move.w  #reBRANCH,(a2)+
214:                    move.w  d0,(a2)+
215:                    bra     orloop
216:
```

```
217: orbreak:          move.l  a1,d2              *未解決だった
218:                   sub.l   a2,d2              *  reBINDの飛び先を格納
219:                   move.w  d2,-(a2)           *
220: orretn:           rts
221:
222: *
223: *         r1・r2・r3・ ... のコンパイル
224: *
225: conccomp:
226: concloop:         bsr     nextchar           *文法要素を1個取り出す
227:                   move.w  jumptable(pc,d0),d2
228:                   jsr     jumptable(pc,d2) *対応ルーチンを呼ぶ
229:                           *reTERM, reRPAREN, reVBARに出会ったら
230:                           *          ここには戻ってこない
231:                   bra     concloop
232: *
233: concbreak:        addq.l  #4,sp              *conccompを抜けて
234:                   rts                        *  orcompに戻る
235: *
236: jumptable:
237: x        =        *
238:                   .dc.w   concbreak-x              *reTERM
239:                   .dc.w   comp_literal-x           *reLITERAL
240:                   .dc.w   comp_dot-x               *reDOT
241:                   .dc.w   comp_class-x             *reCLASS
242:                   .dc.w   comp_class-x             *reNCLASS
243:                   .dc.w   comp_caret-x             *reCARET
244:                   .dc.w   comp_doller-x            *reDOLLER
245:                   .dc.w   comp_lbracket-x          *reLBRACKET
246:                   .dc.w   comp_rbracket-x          *reRBRACKET
247:                   .dc.w   comp_group-x             *reLPAREN
248:                   .dc.w   concbreak-x              *reRPAREN
249:                   .dc.w   comp_refer-x             *reREFER
250:                   .dc.w   literal_asterisk-x       *reASTERISK
251:                   .dc.w   literal_plus-x           *rePLUS
252:                   .dc.w   literal_qmark-x          *reQMARK
253:                   .dc.w   concbreak-x              *reVBAR
254:
255: *
256: *         「^」「$」「¥<」「¥>」
257: *
258: comp_caret:
259: comp_doller:
260: comp_lbracket:
261: comp_rbracket:
262:                   PUT1    d0
263:                   rts
264:
265: *
266: *         「¥( ~ ¥)」
267: *
268: comp_group:
269:                   movea.l a1,a3
270: .ifdef  __GROUPING__
271:                   addq.w  #1,d5
272:                   cmpi.w  #10+1,d5
273:                   bcs     compgroup0
274:                   moveq.l #10,d5
275: compgroup0:       PUT2    d0,d5              *reLPAREN, n
276: .endif
277:                   movem.l a2/a3,-(sp)        *「¥(~¥)」の中身を
278:                   bsr     orcomp             *  コンパイル
279:                   movem.l (sp)+,a2/a3        *
280:
281:                   cmpi.w  #reRPAREN,d0       *「¥)」で終わっている?
282:                   bne     missingparen       *「¥)」が閉じていない
283: .ifdef  __GROUPING__
284:                   move.w  2(a3),d1
285:                   PUT2    d0,d1              *reRPAREN, n
286:                   bset.l  d1,d4
287: .endif
288:                   bra     comp_repeat
289:
290: *
291: *         「.」
292: *
293: comp_dot:
294:                   movea.l a1,a3
295:                   PUT1    d0                 *reDOT
296:                   bra     comp_repeat
297:
298: *
299: *         「¥1」~「¥9」
300: *
301: comp_refer:
302: .ifdef  __GROUPING__
303:                   btst.l  d1,d4
304:                   bne     comp_literal
305:
306:                   addi.b  #'0',d1
307:                   moveq.l #reLITERAL,d0
308:                   bra     comp_literal
309: .endif
310:
311: *
312: *         通常文字
313: *
314: literal_qmark:
315:                   moveq.l #'?',d1            *単独で現れた「?」は
316:                   bra     demote             *  通常文字として扱う
317: *
318: literal_plus:
319:                   moveq.l #'+',d1            *単独で現れた「+」は
320:                   bra     demote             *  通常文字として扱う
321: *
322: literal_asterisk:
323:                   moveq.l #'*',d1            *単独で現れた「*」は
324: *                 bra     demote             *  通常文字として扱う
325: demote:           moveq.l #reLITERAL,d0
326: *                 bra     compliteral
```

```
327: *
328: comp_literal:
329:                   movea.l a1,a3
330:                   PUT2    d0,d1              *reLITERAL, char
331:                   *
332: comp_repeat:      bsr     nextchar           *直後に繰り返し記号があれば
333:                   cmpi.w  #reASTERISK,d0     *  対応するルーチンへ飛ぶ
334:                   beq     comp_asterisk      *
335:                   cmpi.w  #rePLUS,d0         *
336:                   beq     comp_plus          *
337:                   cmpi.w  #reQMARK,d0        *
338:                   beq     comp_qmark         *
339:                   PBACK                      *読みすぎた文字を押し戻す
340:                   rts
341:
342: *
343: *         「[~]」「[^~]」
344: *
345: comp_class:
346:                   movea.l a1,a3
347:                   cmpi.b  #'^',(a0)          *「[^」?
348:                   bne     pushclass          *
349:                   moveq.l #reNCLASS,d0       *
350:                   addq.l  #1,a0              *
351:
352: pushclass:        PUT2    d0,d0              *reCLASS | reNCLASS
353:                                              *(2ワード目はダミー)
354:                   subi.w  #256/8,d6          *256ビットの
355:                   bcs     toolong            *  テーブルを用意する
356:
357:                   movea.l a1,a5              *a5 = ビットのテーブル
358:                   move.l  d3,a6              *d3を待避
359:
360:                   moveq.l #0,d0              *ビットのテーブルを
361:                   moveq.l #256/8/4-1,d1      *  初期化
362: clrfieldloop:     move.l  d0,(a1)+           *
363:                   dbra    d1,clrfieldloop    *
364:
365:                   move.b  (a0)+,d0           *先頭では
366:                   bne     classloopent       *  「]」も通常文字扱い
367:                   bra     missingbracket     *
368: classloop:        moveq.l #0,d0              *d0に下限を得る
369:                   move.b  (a0)+,d0           *
370:                   beq     missingbracket     *
371:                   cmpi.b  #']',d0            *文字クラスの終端?
372:                   beq     classbreak         *
373: classloopent:     BHAN    d0,class0          *
374:                   lsl.w   #8,d0              *
375:                   move.b  (a0)+,d0           *
376:                   beq     missingbracket     *
377:
378: class0:           move.w  d0,d1              *仮に 下限 = 上限
379:
380:                   cmpi.b  #'-',(a0)          *範囲の指定?
381:                   bne     class2             *
382:                   cmpi.b  #']',1(a0)         *
383:                   beq     class2             *
384:                   addq.l  #1,a0              *d1に上限を得る
385:                   moveq.l #0,d1              *
386:                   move.b  (a0)+,d1           *
387:                   beq     missingbracket     *
388:                   BHAN    d1,class1          *
389:                   lsl.w   #8,d1              *
390:                   move.b  (a0)+,d1           *
391:                   beq     missingbracket     *
392:
393: class1:           cmp.w   d0,d1              *下限≦上限を保証する
394:                   bcc     class2             *
395:                   exg.l   d0,d1              *
396:
397: class2:           tst.w   d1                 *上限≦00ffhなら
398:                   bpl     han2han            *  半角~半角
399:                   tst.w   d0                 *下限>00ffhなら
400:                   bmi     zen2zen            *  全角~全角
401:
402: han2zen:          PUT2    #$8000,d1          *半角~全角
403:                   swap.w  d0                 *d0.wを待避
404:                   move.w  #$0080,d0          *
405:                   lsr.w   #8,d1              *
406:                   bsr     setbitfield        *
407:                   swap.w  d0                 *d0.wを復帰
408:                   move.w  #$00ff,d1          *
409:                   *
410: han2han:          bsr     setbitfield        *半角~半角
411:                   bra     classloop
412:
413: zen2zen:          PUT2    d0,d1              *全角~全角
414:                   lsr.w   #8,d0              *
415:                   lsr.w   #8,d1              *
416:                   bsr     setbitfield        *
417:                   bra     classloop
418:
419: classbreak:       move.l  a6,d3              *d3を復帰
420:                   PUT1    #reTERM            *「]」相当
421:
422:                   move.l  a1,d0              *「[」~「]」までの
423:                   sub.l   a3,d0              *  長さを格納
424:                   subq.w  #4,d0              *
425:                   move.w  d0,2(a3)           *
426:
427:                   bra     comp_repeat
428:
429: *
430: *         「[~]」「[^~]」用のビットテーブルを作成する
431: *
432: setbitfield:
433:                   moveq.l #7,d2              *
434:                   and.b   d0,d2              *d2 = 下限のビット位置
435:                   lsr.w   #3,d0              *d0 = 下限のバイト位置
436:
```

▶カゼ薬を飲んで，オーバーテイクの曲を口ずさみながらカートを運転すると，死ぬかもしれないのでやめましょう。僕はやりました……。　　村上　元章(17)静岡県

```
437:                moveq.l #7,d3           *
438:                and.b   d1,d3           *d3 = 上限のビット位置
439:                lsr.w   #3,d1           *d1 = 上限のバイト位置
440:
441:                cmp.w   d1,d0
442:                bne     setbitfield0
443:
444:                        *d0 = d1 のとき
445:                move.b  lboundtable(pc,d2),d2
446:                and.b   uboundtable(pc,d3),d2
447:                or.b    d2,0(a5,d0)
448:                bra     setbfretn
449:
450: setbitfield0:  tst.b   d2
451:                beq     setbitfield1
452:                        *下限の8ビット未満の余り
453:                move.b  lboundtable(pc,d2),d2
454:                or.b    d2,0(a5,d0)
455:                addq.w  #1,d0
456:
457: setbitfield1:  sub.w   d0,d1           *中間
458:                beq     setbitfield3    *
459:                subq.w  #1,d1           *
460: setbitfield2:  st.b    0(a5,d0)        *8ビットすべて埋める
461:                addq.w  #1,d0           *
462:                dbra    d1,setbitfield2 *
463:
464:                        *上限の8ビット未満の余り
465: setbitfield3:  move.b  uboundtable(pc,d3),d3
166:                or.b    d3,0(a5,d0)
467:
468: setbfretn:     rts
469: *
470: lboundtable:   .dc.b   %11111111       *下限のマスク
471:                .dc.b   %11111110
472:                .dc.b   %11111100
473:                .dc.b   %11111000
474:                .dc.b   %11110000
475:                .dc.b   %11100000
476:                .dc.b   %11000000
477:                .dc.b   %10000000
478: *
479: uboundtable:   .dc.b   %00000001       *上限のマスク
480:                .dc.b   %00000011
481:                .dc.b   %00000111
482:                .dc.b   %00001111
483:                .dc.b   %00011111
484:                .dc.b   %00111111
485:                .dc.b   %01111111
486:                .dc.b   %11111111
487:
488: *
489: *          「*」「+」「?」
490: *
491: comp_asterisk:
492:                bsr     nextchar        *繰り返しを最適化する
493:                cmpi.w  #reASTERISK,d0  *  r** → r*
494:                beq     comp_asterisk   *
495:                cmpi.w  #rePLUS,d0      *  r*+ → r*
496:                beq     comp_asterisk   *
497:                cmpi.w  #reQMARK,d0     *  r*? → r*
498:                beq     comp_asterisk   *
499:                PBACK                   *読みすぎた文字を押し戻す
500:
501:                move.l  a1,d0           *       ┌─┐
502:                sub.l   a3,d0           *       ↓ │
503:                beq     asteriskretn    *  →・→r→・next
504:                INSERT  4               *     │     ↑
505:                addq.w  #4,d0           *     └─────┘
506:                move.w  #reBRANCH,(a3)+ *
507:                move.w  d0,(a3)+        *
508:                neg.w   d0              *
509:                PUT2    #reMAIN,d0      *
510: asteriskretn:  rts
511: *
512: comp_plus:
513:                bsr     nextchar        *繰り返しを最適化する
514:                cmpi.w  #reASTERISK,d0  *  r+* → r*
515:                beq     comp_asterisk   *
516:                cmpi.w  #rePLUS,d0      *  r++ → r+
517:                beq     comp_plus       *
518:                cmpi.w  #reQMARK,d0     *  r+? → r*
519:                beq     comp_asterisk   *
520:                PBACK                   *読みすぎた文字を押し戻す
521:
522:                move.l  a3,d0           *    ┌─┐
523:                sub.l   a1,d0           *    ↓ │
524:                beq     plusretn        *  →r→・→next
525:                subq.w  #4,d0           *
526:                PUT2    #reMAIN,d0      *
527: plusretn:      rts
528: *
529: comp_qmark:
530:                bsr     nextchar        *繰り返しを最適化する
531:                cmpi.w  #reASTERISK,d0  *  r?* → r*
532:                beq     comp_asterisk   *
533:                cmpi.w  #rePLUS,d0      *  r?+ → r*
534:                beq     comp_asterisk   *
535:                cmpi.w  #reQMARK,d0     *  r?? → r?
536:                beq     comp_qmark      *
537:                PBACK                   *読みすぎた文字を押し戻す
538:
539:                move.l  a1,d0           *  →・→r→next
540:                sub.l   a3,d0           *    │      ↑
541:                beq     qmarkretn       *    └──────┘
542:                INSERT  4               *
543:                move.w  #reBRANCH,(a3)+ *
544:                move.w  d0,(a3)+        *
545: qmarkretn:     rts
546:
547: *
548: *          つぎの文字を得る
549: *
550: nextchar:
551:                move.w  d3,d0           *押し戻された文字が
552:                beq     nextchar0       *  あれば
553:                swap.w  d3              *  それを返す
554:                move.w  d3,d1           *
555:                moveq.l #0,d3           *
556:                rts
557: *
558: nextchar0:     moveq.l #0,d1           *1バイト取り出す
559:                move.b  (a0)+,d1        *
560:                beq     eos             *
561:                BHAN    d1,nextcharhan  *
562: zenliteral:    lsl.w   #8,d1           *2バイト文字
563:                move.b  (a0)+,d1        *
564:                beq     eos             *
565: literal:       moveq.l #reLITERAL,d0   *
566:                rts                     *
567: *
568: eos:           subq.l  #1,a0           *(a0) = 00h
569:                moveq.l #reTERM,d0
570:                rts
571: *
572: nextcharhan:   cmpi.b  #'¥',d1         *1バイト文字
573:                beq     getesctype
574:
575:                        *「¥」を伴わないメタ文字?
576:                lea.l   syntaxtable1(pc),a5
577: nextcharloop1: move.w  (a5)+,d0
578:                beq     literal         *違ったら通常文字
579:                cmp.b   d0,d1
580:                bne     nextcharloop1
581:                lsr.w   #8,d0
582:                rts
583: *
584: syntaxtable1:  .dc.b   reDOT,'.'       *「¥」なしメタ文字の
585:                .dc.b   reASTERISK,'*'  *  テーブル
586:                .dc.b   rePLUS,'+'      *
587:                .dc.b   reQMARK,'?'     *
588:                .dc.b   reCARET,'^'     *
589:                .dc.b   reDOLLER,'$'    *
590:                .dc.b   reCLASS,'['     *
591:                .dc.b   0,0             *
592: *
593: getesctype:    move.b  (a0)+,d1        *d1 =「¥」のつぎの1文字
594:                beq     illegalesc      *
595:                BZEN    d1,zenliteral   *「¥」+全角は通常文字
596:
597: .ifdef  __GROUPING__
598:                ISDIGIT d1              *¥1～¥9?
599:                beq     notdigit        *
600:                moveq.l #'0',d0         *
601:                cmp.b   d0,d1           *
602:                beq     literal         *
603:                sub.b   d0,d1           *
604:                moveq.l #reREFER,d0     *
605:                rts
606: notdigit:
607: .endif
608:                        *「¥」を伴うメタ文字?
609:                lea.l   syntaxtable2(pc),a5
610: nextcharloop2: move.w  (a5)+,d0
611:                beq     getescchar      *違った
612:                cmp.b   d0,d1
613:                bne     nextcharloop2
614:                lsr.w   #8,d0
615:                rts
616: *
617: syntaxtable2:  .dc.b   reLPAREN,'('    *「¥」つきメタ文字の
618:                .dc.b   reRPAREN,')'    *  テーブル
619:                .dc.b   reVBAR,'|'      *
620:                .dc.b   reLBRACKET,'<'  *
621:                .dc.b   reRBRACKET,'>'  *
622:                .dc.b   0,0             *
623: *
624: getescchar:             *C風のエスケープシーケンス?
625:                cmpi.b  #'x',d1
626:                beq     hexesc
627:                lea.l   esctable(pc),a5
628: nextcharloop3: move.w  (a5)+,d0
629:                beq     literal         *違ったら通常文字
630:                cmp.b   d0,d1
631:                bne     nextcharloop3
632:                move.b  -2(a5),d1
633:                moveq.l #reLITERAL,d0
634:                rts
635: *
636: esctable:      .dc.b   'G'-'@','a'     * ¥a = ^G
637:                .dc.b   'H'-'@','b'     * ¥b = ^H
638: *              .dc.b   '['-'@','e'     * ¥e = ^[
639:                .dc.b   'L'-'@','f'     * ¥f = ^L
640:                .dc.b   'J'-'@','n'     * ¥n = ^J
641:                .dc.b   'M'-'@','r'     * ¥r = ^M
642:                .dc.b   'I'-'@','t'     * ¥t = ^I
643:                .dc.b   'K'-'@','v'     * ¥v = ^K
644:                .dc.b   0,0
645: *
646: hexesc:                 *16進エスケープシーケンス
647:                moveq.l #0,d1
648:                move.b  (a0)+,d0
649:                beq     illegalhexesc
650:                ISHEX   d0
651:                beq     illegalhexesc
652:
653: hexescloop:    ori.b   #$20,d0
654:                subi.b  #'0',d0
655:                cmpi.b  #10,d0
656:                bcs     addhex
```

▶石上氏へ。私も「卒論が……」をよく使いました。そのほかのバージョンとして「修論で……」「学会で……」「データがまだない」などがあります。　　加藤　信之(22)東京都

```
657:                    subi.b   #'a'-'0'-10,d0
658: addhex:            lsl.w    #4,d1
659:                    add.b    d0,d1
660:
661:                    move.b   (a0)+,d0
662:                    beq      hexescretn
663:                    ISHEX    d0                 *いちおう ANSI C 風に
664:                    bne      hexescloop         *  無限長の16進数を許す
665:
666: hexescretn:        subq.l   #1,a0
667:                    moveq.l  #reLITERAL,d0
668:                    rts
669:
670: *
671: *          エラー
672: *
673: missingbracket: lea.l    missbrackmes(pc),a0
674:                    bra      errorretn
675: *
676: unmatchedparen: lea.l    unmatparenmes(pc),a0
677:                    bra      errorretn
678: *
679: missingparen:
680:                    lea.l    missparenmes(pc),a0
681:                    bra      errorretn
682: *
683: toolong:           lea.l    toolongmes(pc),a0
684:                    bra      errorretn
685: *
686: illegalesc:        lea.l    illegalescmes(pc),a0
687:                    bra      errorretn
688: *
689: illegalhexesc:     lea.l    illhexescmes(pc),a0
690:                    bra      errorretn
691: *
692: nomem:             lea.l    nomemmes(pc),a0
693:                    *
694: errorretn.         movea.l d7,sp              *spを修正
695:                    moveq.l #-1,d0             *N=1
696:                    bra      compretn
697: *
698: nomemmes:          .dc.b    'メモリ不足です',0
699: missbrackmes:      .dc.b    '[ に対応する ] がありません',0
700: unmatparenmes:     .dc.b    '¥) に対応する ¥( がありません',0
701: missparenmes:      .dc.b    '¥( に対応する ¥) がありません',0
702: illegalescmes:     .dc.b    '¥ の後ろに文字がありません',0
703: illhexescmes:      .dc.b    '¥x の後ろに16進数がありません',0
704: toolongmes:        .dc.b    'パターンが長すぎます',0
705:
706:                    .end
```

リスト4 REEXEC.S

```
  1: *          正規表現によるパターン照合(実行部)
  2:
  3:                    .include        re.h
  4:                    .include        myctype.h
  5:                    .include        const.h
  6:
  7:                    .xdef    re_exec
  8:                    .xdef    re_regs
  9:                    .xref    bm_exec
 10:                    .xref    ctypetable
 11: *
 12: __CTYPETABLE__     reg      a5
 13: CWORD              equ      CDIGIT.or.CALPHA
 14:
 15: *
 16: *          デクの先頭にデータを追加する
 17: *
 18: PUSH               macro    dat
 19:                    local    skip
 20:                    cmp.l    a3,d6
 21:                    bcs      skip
 22:                    movea.l d7,a3
 23: skip:              move.l   dat,-(a3)
 24:                    .endm
 25:
 26: *
 27: *          デクの先頭からデータを取り出す
 28: *
 29: POP                macro    dreg
 30:                    local    skip
 31:                    cmpa.l   d7,a3
 32:                    bcs      skip
 33:                    movea.l d6,a3
 34: skip:              move.l   (a3)+,dreg
 35:                    .endm
 36:
 37: *
 38: *          デクの末尾にデータを追加する
 39: *
 40: PUT                macro    dat
 41:                    local    skip
 42:                    cmpa.l   d7,a4
 43:                    bcs      skip
 44:                    movea.l d6,a4
 45: skip:              move.l   dat,(a4)+
 46:                    .endm
 47:
 48: *
 49: *          照合成功位置の記録用領域
 50: *
 51: re_regs:           .ds.b    SIZEofREREG
 52:
 53: *
 54: *          re_execの引数構造
 55: *
 56:                    .offset 4
 57: *
 58: txt:               .ds.l    1         *テキスト先頭
 59: txtend:            .ds.l    1         *テキスト末尾
 60: pat:               .ds.l    1         *コンパイル済みパターン
 61: work:              .ds.l    1         *作業用領域先頭
 62: workend:           .ds.l    1         *作業用領域末尾
 63: lastchar:          .ds.w    1         *LF なら txtは行頭
 64: *
 65:                    .text
 66:                    .even
 67: *
 68: re_exec:
 69: SAVREGS =          d1-d7/a1-a6
 70: SAVSIZ  =          (7+6)*4
 71:                    movem.l SAVREGS,-(sp)
 72:
 73:                    movem.l txt+SAVSIZ(sp),d3-d7
 74:                                               *d3 = テキスト先頭
 75:                                               *d4 = テキスト末尾
 76:                                               *d5 = パターン格納領域
 77:                                               *d6 = 作業用領域先頭
 78:                                               *d7 = 作業用領域末尾
 79:                    lea.l    ctypetable,a5     *a5 = 文字種判別テーブル
 80:
 81:                    movea.l d3,a2              *a2 = 照合開始位置
 82:                    move.w   lastchar+SAVSIZ(sp),d2
 83:                    move.l   d4,-(sp)          *(sp) = テキスト末尾
 84:
 85:                    movea.l d5,a0
 86:                    move.w   (a0)+,d3          *d3 = モード
 87:                    move.l   a0,d5
 88:
 89:                    lea.l    re_regs(pc),a0    *照合成功位置を記録する
 90:                    moveq.l #0,d0              *  ワークをクリア
 91:                    clr.l    (a0)+             *
 92:                    clr.l    (a0)+             *
 93:
 94:                    movea.l d6,a3              *a3 = デク先頭ポインタ
 95: *                  lea.l    64(a3),a3         *
 96:                    bra      loop0ent
 97:
 98: *          d0~d1   作業用
 99: *          d2      入力文字
100: *          d3      モード
101: *          d4      テキスト末尾(実際にはスタックに積みっ放し)
102: *          d5      NFA先頭
103: *          d6      デク領域先頭
104: *          d7      デク領域末尾
105: *          a0      テキスト走査ポインタ
106: *          a1      NFAの現在の状態を指すポインタ
107: *          a2      テキスト側
108: *          a3      デク中有効データ先頭
109: *          a4      デク中有効データ末尾
110: *          a5      文字種判別テーブル
111: *          a6      作業用
112: *          (sp).l  テキスト末尾(bm_exec用)
113:
114: loop0:             tst.b    d3                *モードの bit7 = 1 なら
115:                    bpl      notbolmode        *  次行先頭までスキップ
116:                                               *
117:                    moveq.l #LF,d0             *
118:                    moveq.l #0,d2              *
119: searchbolloop:     move.b   (a2)+,d2          *
120:                    beq      nomatch           *
121:                    cmp.b    d0,d2             *LF?
122:                    bne      searchbolloop     *
123:                    bra      loop0ent          *
124:
125: notbolmode:        moveq.l #0,d2              *照合開始位置を
126:                    move.b   (a2)+,d2          *  1文字ずらす
127:                    beq      nomatch           *     ~
128:                    BHAN     d2,loop0ent       *
129:                    lsl.w    #8,d2             *
130:                    move.b   (a2)+,d2          *
131:                    beq      nomatch           *
132:
133: loop0ent:          movea.l a3,a4              *デクを空にする
134:
135:                    moveq.l #0,d0              *区切りの0をデクにセット
136:                    PUT      d0                *
137:
138:                    movea.l a2,a0              *a0 = 照合開始位置
139:                    move.l   d5,d0             *d0 = NFA先頭
140:
141:                    tst.w    d3                *モードの bit15 = 1 なら
142:                    bpl      loop1ent          *  先頭部分をBM法で探す
143:                                               *
144:                    pea.l    (a0)              *
145:                    jsr      bm_exec           *
146:                    addq.l   #4,sp             *
147:                    bne      nomatch           *見つからなければ即終了
148:                                               *
149:                    movea.l a0,a2              *a0 = a2 = 照合開始位置
150:                    moveq.l #0,d2              *
151:                    move.b   -1(a2),d2         *d2 = 直前の文字
152:                    bra      loop1ent
153:
154: loop1:             POP      d0                *取り得る状態がなければ
155:                    beq      break1            *  このループを抜ける
156:
157: loop2:             movea.l d0,a1              *新たな入力文字によって
158:                    bsr      transit           *  遷移できる状態を
159:                                               *  デク末尾に追加
160: next2:             POP      d0                *区切りに出会うまで
161:                    bne      loop2             *  繰り返す
162:                    PUT      d0                *デク末尾に区切りを移す
```

▶ユンケル飲んで入試を受けたら，休憩時間に鼻血が出た。我ながら情けない。
松本　拓司(19)埼玉県

```
163:
164:                    movea.l d5,a1           *内部フラグをクリア
165:                    bsr     clrflag         *
166:                    bra     next3
167:
168: looplent:
169: loop3:             movea.l d0,a1           *次の文字を読まずに
170:                    bsr     eclosure        *  遷移できる状態を
171:                                            *  デク先頭に追加する
172: next3:             POP     d0              *区切りに出会うまで
173:                    bne     loop3           *  繰り返す
174:                    PUT     d0              *デク末尾に区切りを移す
175:
176: next1:             moveq.l #0,d2           *テキストから
177:                    move.b  (a0)+,d2        *  つぎの1文字を読む
178:                    beq     reacheos        *テキスト末尾に達した
179:
180:                    BHAN    d2,gethanchar
181:                    lsl.w   #8,d2           *2バイト文字
182:                    move.b  (a0)+,d2        *
183:                    bne     loop1           *
184:                    bra     reacheos        *第2バイトが0でも
185:                                            *  テキスト末尾とみなす
186:
187: gethanchar:        cmpi.b  #CR,d2          *
188:                    bhi     loop1           *
189:                    bne     notcr           *
190:                    cmpi.b  #LF,(a0)        *CR+LF?
191:                    bne     loop1           *単独のCRは文字扱い
192:                    bra     reacheol        *行末に達した
193: notcr:             cmpi.b  #LF,d2          *行末に達するまで
194:                    bne     loop1           *  繰り返す
195:
196: reacheos:          subq.l  #1,a0           *(a0) = LF | NUL
197:
198: reacheol:          movea.l d5,a1           *内部フラグをクリア
199:                    bsr     clrflag         *
200:
201:                    move.l  re_regs+reTAIL(pc),d0
202:                                            *照合が得られた?
203:                    bne     match
204:
205:                    movea.l a0,a2           *a2 = 行末
206:                    bra     loop0
207:
208: break1:
209: next0:             movea.l d5,a1           *内部フラグをクリア
210:                    bsr     clrflag         *
211:
212:                    move.l  re_regs+reTAIL(pc),d0
213:                                            *照合が得られた?
214:                    beq     loop0           *  まだならやり直し
215:
216: match:             lea.l   re_regs(pc),a0  *a0 = 結果格納領域
217:                    move.l  a2,reHEAD(a0)   *照合成功位置先頭
218:                    sub.l   a2,d0           *d0 = その文字列長
219:                    moveq.l #0,d1           *Z=1, N=0
220:
221: retn:              addq.l  #4,sp
222:                    movem.l (sp)+,SAVREGS
223:                    rts
224: *
225: fastretn:          addq.l  #4,sp
226:                    bra     break1
227:
228: *
229: *          照合失敗
230: *
231: nomatch:           moveq.l #-1,d0          *Z=0, N=1
232:                    bra     retn
233:
234: *
235: *          つぎの文字を読まずに遷移できるすべての状態を探す
236: *
237: eclosureloop:      move.l  a1,d0
238: eclosure:          move.w  (a1)+,d1        *d1 = 構文要素内部コード
239:                    bmi     eclosureretn    *負ならすでに検査した
240:                    st.b    -2(a1)          *検査が重複しないように
241:                                            *  フラグを立てる
242:                    move.w  ejumptable(pc,d1),d1
243:                    jmp     ejumptable(pc,d1)
244: eclosureretn:      rts
245: *
246: ejumptable:
247: x          =       *
248:                    .dc.w   exec_term-x     *reTERM
249:                    .dc.w   through-x       *reLITERAL
250:                    .dc.w   through-x       *reDOT
251:                    .dc.w   through-x       *reCLASS
252:                    .dc.w   through-x       *reNCLASS
253:                    .dc.w   exec_caret-x    *reCARET
254:                    .dc.w   exec_doller-x   *reDOLLER
255:                    .dc.w   exec_lbracket-x *reLBRACKET
256:                    .dc.w   exec_rbracket-x *reRBRACKET
257:                    .dc.w   exec_lparen-x   *reLPAREN (未対応)
258:                    .dc.w   exec_rparen-x   *reRPAREN (未対応)
259:                    .dc.w   exec_refer-x    *reREFER (未対応)
260:                    .dc.w   error-x         *reASTERISK
261:                    .dc.w   error-x         *rePLUS
262:                    .dc.w   error-x         *reQMARK
263:                    .dc.w   error-x         *reVBAR
264:                    .dc.w   exec_main-x     *reMAIN
265:                    .dc.w   exec_branch-x   *reBRANCH
266:                    .dc.w   exec_bind-x     *reBIND
267: *
268: exec_lparen:
269: exec_rparen:
270: exec_refer:
271: error:             divu.w  #0,d0           *内部エラー
272: *
273: through:           PUT     d0              *つぎの文字を読まないと
274:                    rts                     *  遷移可能かどうかが不明
275: *
276: exec_term:                                 *受理状態
277:                    lea.l   re_regs(pc),a6  *照合成功位置末尾を
278:                    move.l  a0,reTAIL(a6)   *  記録
279:
280:                    btst.l  #reFASTRETN_BIT,d3
281:                                            *高速終了モードなら
282:                    bne     fastretn        *  照合成功ですぐ帰る
283:                    rts                     *
284: *
285: exec_caret:                                *「^」
286:                    cmpi.w  #LF,d2          *直前の文字がLF?
287:                    beq     eclosureloop    *
288:                    rts                     *
289: *
290: exec_doller:                               *「$」
291:                    move.b  (a0),d0         *直後の文字コードが00h?
292:                    beq     eclosureloop    *
293:                    cmpi.b  #LF,d0          *  LF?
294:                    beq     eclosureloop    *
295:                    cmpi.b  #CR,d0          *  CR+LF?
296:                    bne     dollerretn      *
297:                    cmpi.b  #LF,1(a0)       *
298:                    beq     eclosureloop    *
299: dollerretn:        rts                     *
300: *
301: exec_lbracket:                             *「¥<」
302:                    tst.w   d2              *直前の文字が
303:                    bmi     lbracketretn    *  語の構成文字でなく
304:                    moveq.l #CWORD,d0       *
305:                    CTEST   d0,d2           *
306:                    bne     lbracketretn    *
307:                    moveq.l #0,d1           *直後の文字が
308:                    move.b  (a0),d1         *  語の構成文字なら
309:                    moveq.l #CWORD,d0       *
310:                    CTEST   d0,d1           *
311:                    bne     eclosureloop    *一致
312: lbracketretn:      rts                     *
313: *
314: exec_rbracket:                             *「¥>」
315:                    tst.w   d2              *直前の文字が
316:                    bmi     rbracket0       *  語の構成文字で
317:                    moveq.l #CWORD,d0       *
318:                    CTEST   d0,d2           *
319:                    beq     rbracketretn    *
320: rbracket0:         moveq.l #0,d1           *直後の文字が
321:                    move.b  (a0),d1         *  語の構成文字でなければ
322:                    moveq.l #CWORD,d0       *
323:                    CTEST   d0,d1           *
324:                    beq     eclosureloop    *一致
325: rbracketretn:      rts                     *
326: *
327: exec_main:
328: exec_branch:                               *枝分かれ
329:                    movea.w (a1)+,a6        *
330:                    adda.l  a1,a6           *a6 = 分岐先
331:                    tst.b   (a6)            *分岐先がチェック済み
332:                    bmi     eclosureloop    *  でなければ
333:                    PUSH    a6              *  デク先頭に押し込む
334:                    bra     eclosureloop    *
335: *
336: exec_bind:                                 *無条件ジャンプ
337:                    movea.w (a1)+,a6        *
338:                    adda.l  a6,a1           *a1 = 飛び先
339:                    bra     eclosureloop    *
340:
341: *
342: *          新たな入力文字によって遷移できるかどうか調べる
343: *
344: transit:
345:                    move.w  #$00ff,d1       *フラグをマスク
346:                    and.w   (a1)+,d1        *d1 = 構文要素内部コード
347:                    move.w  jumptable(pc,d1),d1
348:                    jmp     jumptable(pc,d1)
349: *
350: jumptable:
351: x          =       *
352:                    .dc.w   error-x         *reTERM
353:                    .dc.w   exec_literal-x  *reLITERAL
354:                    .dc.w   exec_dot-x      *reDOT
355:                    .dc.w   exec_class-x    *reCLASS
356:                    .dc.w   exec_nclass-x   *reNCLASS
357:                    .dc.w   error-x         *reCARET
358:                    .dc.w   error-x         *reDOLLER
359:                    .dc.w   error-x         *reLBRACKET
360:                    .dc.w   error-x         *reRBRACKET
361:                    .dc.w   error-x         *reLPAREN
362:                    .dc.w   error-x         *reRPAREN
363:                    .dc.w   error-x         *reREFER
364:                    .dc.w   error-x         *reASTERISK
365:                    .dc.w   error-x         *rePLUS
366:                    .dc.w   error-x         *reQMARK
367:                    .dc.w   error-x         *reVBAR
368:                    .dc.w   error-x         *reMAIN
369:                    .dc.w   error-x         *reBRANCH
370:                    .dc.w   error-x         *reBIND
371: *
372: exec_class:                                *「[~]」
373:                    moveq.l #7,d0           *
374:                    move.w  d2,d1           *
375:                    bmi     zenclass        *
376:                                            *
377:                    and.w   d1,d0           *1バイト文字のときは
378:                    lsr.w   #3,d1           *  ビットテーブルを参照
379:                    btst.b  d0,2(a1,d1)     *
380:                    beq     classretn       *
381: matchclass:        movea.w (a1)+,a6        *
382:                    adda.l  a6,a1           *
```

▶ナショナル「モミモミ」を揉み揉みした。　　　大久保　明弘(20)岩手県

```
383:                    bra     putit
384: *
385: zenclass:           lsr.w   #8,d1               *2バイト文字のときは
386:                    and.w   d1,d0               *  上位バイトについて
387:                    lsr.w   #3,d1               *  ビットテーブルを参照し
388:                    btst.b  d0,2(a1,d1)         *  大まかに判定
389:                    beq     classretn           *
390:
391:                    lea.l   256/8+2(a1),a6      *それから
392:                    bra     classloopent        *  文字リストを探索して
393: classloop:          cmp.w   d0,d2               *  厳密に判定
394:                    bcs     classnext           *
395:                    cmp.w   (a6),d2             *
396:                    bls     matchclass          *
397: classnext:          addq.l  #2,a6               *
398: classloopent:       move.w  (a6)+,d0            *
399:                    bne     classloop           *
400: classretn:          rts                         *
401: *
402: exec_nclass:                                    *「[^~]」
403:                    moveq.l #7,d0               *
404:                    move.w  d2,d1               *
405:                    bmi     zennclass           *
406:                                                *
407:                    and.w   d1,d0               *1バイト文字
408:                    lsr.w   #3,d1               *
409:                    btst.b  d0,2(a1,d1)         *
410:                    beq     matchclass          *
411:                    rts                         *
412: *
413: zennclass:          lsr.w   #8,d1               *2バイト文字
414:                    and.w   d1,d0               *
415:                    lsr.w   #3,d1               *
416:                    btst.b  d0,2(a1,d1)         *
417:                    beq     matchclass          *
418:                                                *
419:                    lea.l   256/8+2(a1),a1      *
420:                    bra     nclassloopent       *
421: nclassloop:         cmp.w   d0,d1               *
422:                    bcs     nclassnext          *
423:                    cmp.w   (a1)+,d1            *
424:                    bhi     nclassloopent       *
425:                    rts                         *
426: nclassnext:         addq.l  #2,a1               *
427: nclassloopent:      move.w  (a1)+,d0            *
428:                    bne     nclassloop          *
429: *                  bra     putit               *
```

```
430: *
431: exec_dot:                                       *「.」
432: putit:              PUT     a1                  *遷移できたので
433:                    rts                         *  デク末尾に追加
434: *
435: exec_literal:                                   *通常の文字
436:                    cmp.w   (a1)+,d2            *
437:                    beq     putit               *
438:                    rts                         *
439:
440: *
441: *          内部フラグをクリアする(d0=0で呼ぶこと)
442: *
443: skipword:           addq.l  #2,a1
444: clrflag:
445: clrflagloop:        move.b  d0,(a1)             *フラグをクリア
446:                    move.w  (a1)+,d1            *d1 = 構文要素内部コード
447:                    move.w  cjumptable(pc,d1),d1
448:                    jmp     cjumptable(pc,d1)
449: skipclass:          movea.w (a1)+,a6
450:                    adda.l  a6,a1
451:                    bra     clrflagloop
452: clrflagretn:        rts
453: *
454: cjumptable:
455: x          =        *
456:                    .dc.w   clrflagretn-x       *reTERM
457:                    .dc.w   skipword-x          *reLITERAL
458:                    .dc.w   clrflagloop-x       *reDOT
459:                    .dc.w   skipclass-x         *reCLASS
460:                    .dc.w   skipclass-x         *reNCLASS
461:                    .dc.w   clrflagloop-x       *reCARET
462:                    .dc.w   clrflagloop-x       *reDOLLER
463:                    .dc.w   clrflagloop-x       *reLBRACKET
464:                    .dc.w   clrflagloop-x       *reRBRACKET
465:                    .dc.w   skipword-x          *reLPAREN
466:                    .dc.w   skipword-x          *reRPAREN
467:                    .dc.w   skipword-x          *reREFER
468:                    .dc.w   error-x             *reASTERISK
469:                    .dc.w   error-x             *rePLUS
470:                    .dc.w   error-x             *reQMARK
471:                    .dc.w   error-x             *reVBAR
472:                    .dc.w   skipword-x          *reMAIN
473:                    .dc.w   skipword-x          *reBRANCH
474:                    .dc.w   skipword-x          *reBIND
475:
476:                    .end
```

リスト5 RETEST.S

```
  1: *          re_comp, re_execの動作試験
  2:
  3:            .include        doscall.mac
  4: *          .include        fefunc.h
  5:            .include        re.h
  6:            .include        const.h
  7: *
  8: FPACK      macro   callno
  9:            .dc.w   callno
 10:            .endm
 11: *
 12: __LTOS     equ     $fe11
 13: *
 14:            .xref   re_comp
 15:            .xref   re_exec
 16:            .xref   re_regs
 17: *
 18: ent:
 19:            lea.l   inisp,sp
 20:
 21:            lea.l   linbuf,a3           *行バッファ初期化
 22:            st.b    (a3)                *
 23:
 24:            move.w  #ROPEN,-(sp)        *ファイルオープン
 25:            pea.l   1(a2)               *
 26:            DOS     _OPEN               *
 27: *          addq.l  #6,sp               *
 28:            move.w  d0,d7               *
 29:            bmi     done                *
 30:
 31:            pea.l   16*1024.w           *ファイル読み込み
 32:            lea.l   buff,a1             *
 33:            move.b  #LF,(a1)+           *a1 = テキスト先頭
 34:            pea.l   (a1)                *
 35:            move.w  d7,-(sp)            *
 36:            DOS     _READ               *
 37:            lea.l   0(a1,d0.l),a2       *a2 = テキスト末尾
 38:            DOS     _CLOSE              *
 39: *          lea.l   10(sp),sp           *
 40:
 41:            cmpi.b  #EOF,-(a2)          *ファイル末のEOFを
 42:            beq     deleof              *  削除
 43:            addq.l  #1,a2               *
 44: deleof:    sf.b    (a2)                *
 45:
 46:            cmpi.b  #LF,-1(a2)          *テキスト末に
 47:            beq     loop0               *  改行がなければ
 48:            move.b  #CR,(a2)+           *  補う
 49:            move.b  #LF,(a2)+           *
 50:            sf.b    (a2)                *
 51:
 52: loop0:     pea.l   prompt(pc)          *正規表現の入力
 53:            DOS     _PRINT              *
 54:            pea.l   (a3)                *
 55:            DOS     _GETS               *
 56:            pea.l   crlfms(pc)          *
 57:            DOS     _PRINT              *
 58:
```

```
 59:            pea.l   cpatend             *正規表現を
 60:            pea.l   cpat                *  コンパイル
 61:            pea.l   2(a3)               *
 62:            bsr     re_comp             *
 63:            lea.l   12+12(sp),sp        *
 64:            bmi     error               *
 65:
 66: *          ori.w   #reFASTRETN,(a0)
 67:
 68:            moveq.l #0,d2               *d2 = 一致行数カウンタ
 69:
 70:            move.w  #LF,-(sp)           *照合
 71:            pea.l   dequeend            *
 72:            pea.l   deque               *
 73:            pea.l   (a0)                *
 74:            movem.l a1/a2,-(sp)         *
 75: loop1:     bsr     re_exec             *
 76:            bne     nfound              *
 77:
 78:            movea.l reTAIL(a0),a5       *a5 = 一致部分末尾
 79:            movea.l reHEAD(a0),a0       *a0 = 一致部分先頭
 80:            addq.l  #1,d2               *カウンタ++
 81:
 82:            movea.l a0,a4               *行頭を探す
 83: hloop:     cmpi.b  #LF,-(a4)           *
 84:            bne     hloop               *
 85:            addq.l  #1,a4               *
 86:
 87:            move.b  (a0),d1             *一致部分の
 88:            sf.b    (a0)                *  直前までを出力
 89:            pea.l   (a4)                *
 90:            DOS     _PRINT              *
 91:            addq.l  #4,sp               *
 92:            move.b  d1,(a0)             *
 93:
 94:            pea.l   bstr(pc)            *表示色を変える
 95:            DOS     _PRINT              *
 96:            addq.l  #4,sp               *
 97:
 98:            move.b  (a5),d1             *一致部分を出力
 99:            sf.b    (a5)                *
100:            pea.l   (a0)                *
101:            DOS     _PRINT              *
102:            addq.l  #4,sp               *
103:            move.b  d1,(a5)             *
104:
105:            pea.l   wstr(pc)            *表示色を元に戻す
106:            DOS     _PRINT              *
107:            addq.l  #4,sp               *
108:
109:            movea.l a5,a0               *行末を探す
110: tloop:     cmpi.b  #LF,(a0)+           *
111:            bne     tloop               *
112:
113:            move.b  (a0),d1             *行の残りを出力
114:            sf.b    (a0)                *
115:            pea.l   (a5)                *
116:            DOS     _PRINT              *
```

```
117:          addq.l   #4,sp          *
118:          move.b   d1,(a0)        *
119:
120:          move.l   a0,(sp)        *a0 = つぎの照合開始位置
121:
122:          cmpa.l   a2,a0          *テキスト末に達するまで
123:          bcs      loop1          *  繰り返す
124:
125: nfound:  lea.l    22(sp),sp
126:
127:          lea.l    2(a3),a0       *一致した行数を出力
128:          pea.l    (a0)           *
129:          move.l   d2,d0          *
130:          moveq.l  #10,d1         *
131:          FPACK    __LTOS         *
132:          DOS      _PRINT         *
133:          pea.l    accmes(pc)     *
134:          DOS      _PRINT         *
135:          addq.l   #4+4,sp        *
136:
137:          bra      loop0
138: *
139: done:    DOS      _EXIT
140: *
141: error:   pea.l    (a0)           *エラーメッセージを
142:          DOS      _PRINT         *  出力
143:          pea.l    crlfms(pc)     *
144:          DOS      _PRINT         *
145:          addq.l   #4+4,sp        *
146:          bra      loop0
147: *
148: prompt:  .dc.b    'パターン = ',0
149: accmes:  .dc.b    '行 と一致しました'
150: crlfms:  .dc.b    CR,LF,0
151: bstr:    .dc.b    ESC,'[31m',0
152: wstr:    .dc.b    ESC,'[m',0
153: *
154:          .bss
155:          .even
156: *
157: linbuf:  .ds.b    256+2              *行バッファ
158: cpat:    .ds.w    256                *コンパイル'ed パターン
159: cpatend:                            *
160: deque:   .ds.w    256*4              *re_execの作業用
161: dequeend:                           *
162: buff:    .ds.b    16*1024+4          *テキスト
163: *
164:          .stack
165:          .even
166: *
167:          .ds.l    1024
168: inisp:
169:
170:          .end     ent
```

正規表現のメタ文字

●.

ピリオドは任意の1文字を表し，どのような文字とでも一致する。Human68kのファイル名パターンで使われるワイルドカード文字「?」に似ているが，空文字列とは一致しない。また，行の終端コードとも一致しない。

●¥

直後の文字のメタ文字としての機能を打ち消すエスケープ文字として働く。たとえば，

　¥.

は「.」という文字と一致する。文字「¥」自体を表したい場合は「¥¥」で表す。

例外的に一部のメタ文字は「¥」＋1文字の形で表される場合がある。また，「¥」＋半角英字をC風のエスケープシーケンスとして扱うプログラムも多い。

●[～]，[^～]

「[」と「]」の対は文字クラス（文字の集合）を表し，そのあいだに並べた任意の1文字と一致する。「[」の直後に「^」を置くと，意味が反転し，列挙した文字以外と一致するようになる。ただし，行の終端コードとは一致しない。

たとえば，

　[abcd]

は文字「a」～「d」のいずれかと一致し，

　[^abcd]

は「a」～「d」以外の文字と一致する。「[～]」内では「-」により，文字の範囲を表すこともできるので，上の例は，それぞれ，

　[a-d]

　[^a-d]

とも書ける。

「[～]」中で特別な意味をもつ「-」と「^」，および，「]」を文字クラスに含ませたい場合は，置く位置を工夫することで対処する。「]」の場合は「[」の直後に置く。「-」は「[」の直後か，「]」の直前に置く。また，「^」は「[」の直後以外に置く。したがって，これらの3文字すべてを「[～]」の中に置きたい場合は，

　[]^-]

と書けばよい。

「[～]」中では，そのほかのメタ文字はメタ文字としての意味を失い，通常の文字として扱われる。まれに「¥」だけはエスケープ文字としての意味をもち続ける場合がある。

●^，$

それぞれ，行頭と行末を表し，行頭の文字の直前にある空文字列，行の終端コードの直前にある空文字列と一致する。正規表現の途中に現れた「^」と「$」の解釈は実装方法による。常にメタ文字として扱うか，無効な位置に現れたら通常の文字とみなすか，がありうる。前者の解釈だと，

　abc^def

は，どんな文字列とも一致しなくなる（行の途中に行頭があるわけがない）。後者の解釈なら，見た目どおりの7文字の文字列と一致する。

**●*

直前のパターンの0回以上の繰り返しと一致する。正規表現の先頭など，無意味な位置に現れた場合の解釈はプログラムによる。無視するか，文字「*」自体とみなすか，エラーにするか，がありうる。

●¥(～¥)

「¥(」と「¥)」は正規表現のグルーピングを行い，演算子の優先順位を変えるのに使う。いくらでも入れ子にできるのがふつうだが，できなかったり，回数に制限があったりする場合もある。なお，プログラムによっては，「¥」をつけない「(」と「)」でグルーピングを行い，「¥(」と「¥)」で文字「(」，「)」自体を表すものもある。

＊　　＊　　＊

ここまでに挙げたメタ文字はほぼどんなプログラムでもサポートされている。プログラムによっては，さらに，以下のメタ文字の一部ないしすべてがサポートされる。

●+，?

「+」は直前のパターンの1回以上の繰り返しと一致する。また，「?」は直前のパターンの0～1回の繰り返しと一致する。文脈上，不適切な位置に現れた場合の解釈は「*」に準じる。

プログラムによっては「+」と「?」でその文字自体を表し，繰り返しの意味で使うときには「¥+」，「¥?」で表現する場合がある。

●¥|

論理和を表し，「¥|」を挟んで並べたパターンのいずれかと一致する。

　abc¥|def

は，「abc」か「def」のいずれかと一致する。やはり，プログラムによって「¥|」と「|」の意味が逆になっている場合がある。「^」または「$」と併用した場合の解釈もプログラムによる。

　^abc¥|def$

は，

　¥(^abc¥|def$¥)

つまり，行頭の「abc」か，行末の「def」と解釈される場合が多いが，まれに，

　^¥(abc¥|def¥)$

と解釈するプログラムもある。

●¥<，¥>

それぞれ，単語の先頭と末尾を表す。扱いは「^」と「$」に似ており，単語の先頭の直前，および，末尾の直後にある空文字列と一致する。ここで，単語とは，通常，英数字だけからなる文字列を指すが，日本語対応のプログラムの場合，半角カタカナや全角文字も単語の構成文字として扱うことがあるかもしれない。

●¥{m，n¥}

m～n回の繰り返しを表す。m，nの範囲はふつう0～255程度に制限される。m，nのどちらかは省略でき，省略した場合，

　¥{m,¥}

は，m回以上，

　¥{,n¥}

なら，n回以下，また，

　¥{m¥}

なら，m回ちょうどの繰り返し指定と解釈される。

●¥1，¥2，……，¥9

後方参照を表し，「¥n」は"n（1～9）番目に現れた「¥(～¥)」と一致した文字列"と一致する。たとえば，

　¥([a-z]¥)¥1

は，「aa」や「zz」など，同じ半角英小文字が2つ並んだものと一致する。

　¥(abc¥1¥)

のように，「¥(～¥)」が閉じる前に使った場合の動作は実装による。空文字列とみなすか，とりあえず途中まで一致した文字列とみなすか，あるいは，文字「1」自体として扱うか，エラーにするか，が考えられる。

実は「¥n」は正規表現の定義からは逸脱しており，"拡張"を頭につけたとしても正規表現とは呼べない。が，今回は正規表現の定義を示していないことでもあるし，多少の嘘には目をつぶって，拡張正規表現のなかにくくってしまう。

繁殖するウイルス

その不作ぶりを不況のせいにするかどうかは別問題としまして，バレンタインデーも無事に終わりました。みなさんはいかがでしたか？　ところで，あっちのほうはどうでしたか？　例の「バレンタイン・ウイルス」です。まぁ，ほとんどいないような気もします。

情報処理振興事業協会が発表したところによると，1992年中のコンピュータウイルスによる被害の届け出数は前年の4.4倍の253件に達したのだそうです。253件という数は多いのでしょうか，少ないのでしょうか？

この数字は実際より全然少なくて，本当の数はこれの何倍，もしかしたら10倍以上あるのではないかと僕には思えてなりません。前年の4.4倍という数字にしたって，この1年の間に，感染したらこの協会に届けるということを知った人の増加が当然あるのでしょうから，かなりアバウトな数字といえるでしょう。

かくいう僕自身，あるいは周囲の人々なども，案外とそれらしきものに「感染」する(感染しかけるがワクチンで予防する)のですが，はっきりいって，その協会などに届けたこともないですし，届けたという話を聞いたこともありません。

「それらしき」という言葉を使った理由は，特にMacintoshの場合には，ウイルスが原因なのか，何が原因なのか，はっきりしない場合が意外と多いからです。おっくうだというのがたぶん最大原因でしょうが，このことも，届け出ないひとつの理由ではあると思います。

お役所も動きだすようです。通産省はコンピュータウイルスに対応するために93年度から4カ年計画で「セキュリティ対策システム研究開発事業」を開始するのだそうです。現在のように新しいウイルスが発見されるごとに新しいワクチンを作るというのではなく，暗号技術を使って「あらゆるウイルスを予防する」のです。暗号キー作成，デジタル署名，暗号解読の3つの部分からシステムは構成される見通しだといいます。

あらゆるウイルスに感染しないすばらしいシステムの完成を僕はもちろん待望します。でも，もし愛用しているMacintoshPowerBook(いまこの文章も揺れるバスの中で入力していますが)を使い始めるごとに，暗号キーを入力してくださいとか，デジタル署名をしてくださいとかせがまれるのであれば，それは七面倒くさすぎます。

確かにこのマシンはやわでして，たまには風邪をひいてワクチンを打たなくてはならなくなりますが，いつでもどこでも気軽にサクッと使えるところが身上のパーソナルなコンピュータなのですから。

ウイルスで大喜びの人々

コンピュータウイルスのことを扱ったテレビ番組を見ました。1月19日にNHKで放映されたプライム10「コンピュータウイルスの恐怖　高度情報化社会の落とし穴」です(テレビマンユニオンが作ったいわゆる外注番組です)。例の「冬彦さん」こと佐野史郎氏がコンピュータウイルス研究所の所長という設定で案内役を務めており，楽しめる作りになっていました。

しかし，何よりもこの番組を出色のものにしていたのは，ウイルスをめぐる犯人や仕掛人たちをカメラの前に登場させて，社会現象としてのコンピュータウイルスというものの全体像をくっきりと浮かび上がらせていたことでしょう。

実名入りボカシなしで登場した犯人は，思ったとおりのイメージの，世間慣れしていない明るそうでない感じの(元？)学生でした。彼は，犯意を否定するとともに，「ソフトウェア追求の自由」的なことを主張しているように見えました。

一方，犯人とはいえないでしょうが，灰色であることは間違いなさそうな仕掛人も登場しました。それは，ワクチン会社の経営者です。彼は狼少年のように，あるウイルスの出現を声高に叫び，そしてそのワクチンを売りまくったそうです。そのようなウイルスは実際にはほとんど出現しなかったと述べる証言者も同時に登場していました。

さらに，このような「コンピュータウイルス産業」で儲ける別の人が紹介されます。その人はウイルスに関する本を出版して儲けているのです。単にそれだけなら，何ということはないのですが，その本にはウイルスそのもののソースコードまでついているというのです。ですから，その気になれば読者がウイルスを作ることも容易というわけなのです。

さすがに番組製作者側はその本を広く紹介するのはまずいと判断したのか，出版社とか正式なタイトルなどについては，はっきりとは画面に出しませんでした。NHKも最近防戦一方ですからね。

悪いウイルスが行うことは犯罪行為そのものです。だけれども，単に悪い，だめだというだけでは，その感染を防ぐワクチンを開発する，対策のための本を出版するといった不安ビジネス(最近読んだ佐高信の本にも登場してきた言葉ですが)を助長するだけですし，あるいは，どんなウイルスが出現してもだいじょうぶなように完全防護のセキュリティを開発して結局自らをがんじがらめにする結果になってしまうように思えてならないのです。

コンピュータウイルスの正体

コンピュータウイルスに関する技術的な話が比較的わかりやすくまとめてある論文(参考文献1)があります。ウイルスはマシン語コードであり，それが実行されると自分自身のコードを別のホストプログラム中に埋め込みます。これが感染ですが，基本的には純粋なデータそのものは感染しません。このような自己増殖機能があるプログラムのことをウイルスとここでは定義しています。

感染するだけで何もしないというウイルスもありますが，普通は何かのきっかけ(特定の日付，あるファイルの存在，キー入力の特定のパターンなど)でその本性を現します。たとえば，単にメッセージを画面に出力するような単純なもの，あるファイルの特定のデータを変更するような複雑なもの，さらにディスク全体を消してしまうような破壊的なものなどさまざまです。

ウイルスと混同されやすいものに「ワーム」があります。ワームは独立して走るプログラムであり，ネットワークを通してマシンからマシンへと乗り移るところに特徴があります。ウイルスと大きく異なるのは，ワームは決してほかのプログラムを書き換えたりしないということです。

もともとワームは1982年にファイル空間をきれいにしたり，メールを運んだり，ワー

クステーションの電源を切ったりするために作られた善玉でした。1988年にアメリカで起こった例の「インターネットワーム事件」で登場したような悪意のあるワームは少ないようです。それは作るのが難しいからということです。

コンピュータウイルスの起源は1980年代の初頭，Apple用のものにさかのぼります。厳密に解析し研究したのは南カリフォルニア大学のF.Cohenであり，彼は1985年に「コンピュータウイルス」というタイトルのついた博士論文を書いています(！)。

ウイルス感染の経路としていちばん典型的なのが次のようなものです。

→購入プログラムが(感染しているため)動かない，とソフトハウスなどに持ち込む
→担当者がそのプログラムを試すために動かし，メモリ上にファイルが移る
→あとで調べるために持ち込まれたディスク上のプログラムを別のディスクにコピーする(コピーツールに感染)
→担当者が別の仕事をするためにエディタを開く(エディタに感染)
→担当者が仕事を終え，マシンのスイッチを切る(メモリはきれいになるがコピーツールとエディタは感染したまま)
→ネットワーク経由で別の人がこのプログラムを起動し，その人のマシンも感染

人工生命としてのウイルス

人工生命(Artificial Life)の研究が盛んになりつつあります。C.G.Langtonによれば，人工生命の研究とは，さまざまな生物学的現象から論理的な形式を抽出しようとするものです。生物学的現象とは，たとえば次のようなものを含みます(訳に怪しいところもあるが勘弁)。

生命の起源，分子の自己集結，発生起源(成長，発展，分化)，動物行動学，虫の集落に関するダイナミクス，進化，種化，生態学のダイナミクス，さらには，言語や社会文化的な進化……。

人工生命の研究の結果，計算機上に生命のようなものをシミュレートできたとします。その場合，それがいかに生命に近いかという判定をする必要が生じます。その場合に生命とはいったい何か，何をもって生命と呼ぶのかという定義が必要となります。

ひとつの尺度として，FarmerやBenlinらによると次のような生命の特徴が挙げられます。

・時間空間的パターンであること
・自己生成できること
・自己表現のための情報をもっていること
・物質とエネルギーを新陳代謝できること
・環境と機能的に相互作用すること
・外乱に対しても安定していること
・進化する能力を持っていること
・成長すること

illustration：Haruhisa Yamada

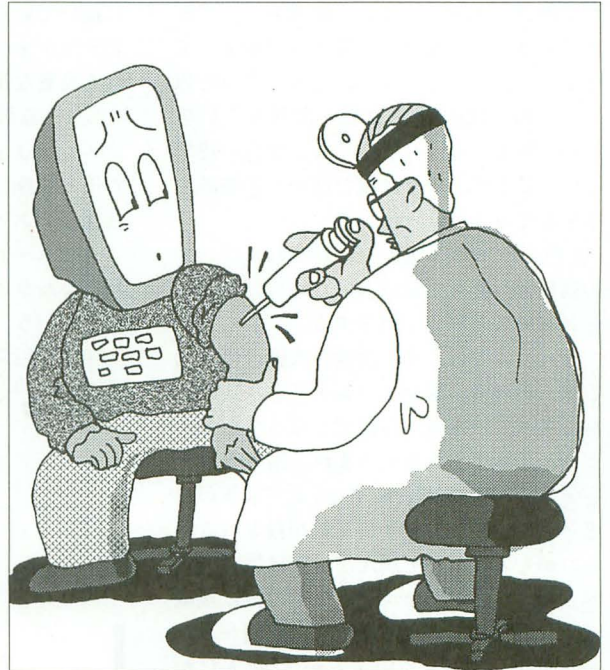
参考文献1に挙げた論文は，コンピュータウイルスの仕組みや働きを述べることに主眼を置いたのではなく，そのタイトルどおり，それを人工生命としてみることはできないのかとの問いかけを行っているものなのです。

この論文のなかで，先の生命の特徴それぞれがコンピュータウイルスに当てはまるかどうかという議論を行い，結論としては，コンピュータウイルスというものはかなり人工生命に近いのだとしています。

この論文で行われている先の特徴に当てはまるかどうかという議論はそれほど厳密なものとは思えません。そのひとつの理由は，深い議論を行おうとすればするほど参考文献2で行われているような哲学的な問題にいきついてしまうからだと思われます。

近未来社会におけるウイルス

バレンタインデーになったら「歯をよくみがいてね！」とただ画面に表示するようなコンピュータウイルスでも，これは他人の計算機資源のなかに勝手に踏み込んでいるのですから，犯罪的行為であることには議論の余地はありません。

しかしNHK番組ではありませんが「恐怖」とか「落とし穴」とかいってウイルスの恐怖をあおりたてる社会現象や，せっかくのパーソナルな計算機を完全防備にしてしまおうとする方向性には少しとまどいを感じてしまいます。そもそも，計算機ネットワーク社会に付随するひとつの現象だと割り切ったほうが，結局トータルとして幸せになれるということはないでしょうか？

風邪の完全な治療薬(予防薬？)を発明すればノーベル賞だとか，よく世間ではいわれますが，風邪があるからこそ，うがいをしたり，体調を整えたり，健康に注意するのでしょう。僕自身ウイルスにファイルを消されるのが怖いこともあって，こまめにバックアップを取るようになってきたのですが，それは同時に「おっとー！　うっかり大事なファイルを消しちゃったよー！」という事故に対する備えにもなっているわけです。

そして，コンピュータウイルスの人工生命としての研究からもし何か得るものがあるのならば，それは犯人たちのつまらないたくらみを超越するようなずいぶんと楽しい話であると思えてならないのです。

参考文献

1）E.H.Spafford, "Computer Viruses-A Form of Artificial Life?", Artificial Life II, Addison Wesley, pp. 727-745(1992).
2）特集――人工生命，現代思想，1991年1月号，pp.37-153(1991).

X-OVER・NIGHT
（クロスオーバーナイト）

［第33話］「X」

TAKAHARA HIDEKI 高原 秀己

昨年後半あたりから米国で，野球帽，Tシャツなど多数の商品に「X」の文字が見られるようになった。

いうまでもなく，米国で昨年末に公開された映画「マルコムX」の関連グッズである。日本でも2月19日から上映が始まっているので，もうご覧になった人も多いだろう。この「Oh!X」とは何の関係もないのだが，奇しくも同じ文字「X」からの連想ということで，今月は観たばかりの映画の話をしてみよう。

まず，「マルコムX」だが，これについての情報をお持ちでない人のために，ちょっとスペースを割いて紹介する。

マルコムXは，第二次世界大戦後の米国の黒人指導者，故マルコム・リトル氏の通称であり，映画は彼の伝記となっている。

1925年ネブラスカ州生まれ。黒人差別を主張する秘密結社K.K.K.により家族が迫害を受け，6歳のときに父親が殺される。それがもとで精神を病んだ母親は施設に収容されてしまい，マルコムはその後は親戚に引き取られて育つ。学校での成績がよかった彼は弁護士を志望するが，「黒人だから大工にでもなりなさい」と教師に反対される。

それやこれやで非行に走り，青年期にはギャングの一味となるが，23歳で逮捕され，刑務所送りになる。そして，ここで黒人イスラム教徒と出会ったことで新しい人生観に目覚め，見事に更生する。

27歳で出所したのちは，デトロイトを拠点に黒人イスラム教徒の一員として布教活動を始める。同じ時期にやはり黒人の人権運動の指導者であったキング牧師が「非暴力，無抵抗」を主張したのに対し，彼は「黒人の正当な人権を獲得するために，白人と戦おう」「やられたらやり返せ」というラジカルでセンセーショナルな呼びかけで「黒い悪魔」などといわれて注目を集め，布教ひとすじの生活をおくる。

だが，その後，所属宗教団体の指導者のスキャンダルが発覚する。嫌気がさした39歳の彼は独立，自らの団体を興して名実ともに指導者としてスタートする。これを機に聖地メッカに巡礼に訪れ，さまざまな人種のイスラム教徒に出会った彼は，白人を攻撃するだけではなく，世界人類全体の平等な社会づくりが必要だという，新しい真理を得る。しかし，元の団体との対立が激化して，1965年に講演中に暗殺されてしまうのである。

映画は，以上の彼の一生を3時間22分にまとめてある。伝えられる彼の一生を忠実に描いているようだ。伝記映画としても，ドラマティックに宗教家の物語を見せてくれる良質な出来となっている。

製作，監督を務めたのは「ニュー・ジャック・シティ」「ジャングル・フィーバー」で黒人映画作家の若手ナンバーワンの座を確保したスパイク・リー。CNNなどのインタビューによると，「どうしてもこの時期にマルコムXの映画を作っておきたかった」という。

1990年以降になって，ようやく黒人監督が黒人俳優を使って黒人社会を描いた映画が製作されるようになってきた。本当に最近のことだ。スパイク・リーのほかにもジョン・シングルトンの「ボーイ・ズン・ザ・フッド」，ビル・デュークの「ディープ・カバー」など多数の作品があり，日本でも比較的大規模に公開されるようになってきている。

こうした一連の作品は，いずれも現在の黒人社会の暗部を断面として描いた作品だ。何本かのエピソードをシャープに描き続けてきたスパイク・リーは，このあたりで黒人問題の根幹を総まとめし，原点に戻った啓蒙をするためにも，黒人の人権確立に生きたマルコムXを題材とした伝記映画を撮ることにしたのだろう。一連の黒人社会を舞台にした黒人映画も，このマルコムXの映画化により，体系が整ったといえる。

この映画，日本人にとっても決して無関係ではない。晩年のマルコムXは日本人に会ったときには「有色人種はすべて共通の立場にある」と教えていたそうだ。共有できる問題ではないし，すべきことでもないが，しかし決して他人事ではない側面も持ち合わせているのだ。

ちょっと重い話題になってしまった。

さて，重いテーマを描いて人々に何かを訴えるのに，映画は非常に大きな力を持っている。しかし，映画の魅力というのはそれだけではない。こうした重さがまったくなく，いわゆるエンターテイメント，娯楽に徹した楽しさというのも映画の醍醐味のひとつである。

そういう映画の代表的な1本として紹介したいのが，3月上旬公開の香港映画「シティハンター」だ。北条司のコミックを原作というか原案とした作品で，主人公の冴場獠にあたる香港人探偵役がジャッキー・チェン。日本企業がスポンサーになっているわけではないが，一連のジャッキー映画と同じゴールデンハーベストの純香港映画だが，後藤久美子が共演するなど，日本のこともかなり意識して作ってある。

この映画では香港と日本が舞台となっている関係もあって，さまざまな「和中折衷」模様がみられた。なかでも，とんねるずの「ガラガラへびがやってきた」を中国語カバーで歌い踊るシーンが妙にハマっていたり，ストリートファイターIIをジャッキーが実演してしまう場面がやけに斬新だったことなど，いろいろ楽しめた。

香港映画は世界で唯一，無国籍映画の楽しさを売りにした映画作りを進めているのだが，ここにコミックの世界やコンピュータゲームの世界を加えると，ますますその色が濃くなって不思議な映像となることがわかったのは，大きな収穫だった。かつて「孔雀王」でも実験されたことだが，あのときは日本資本の映画であり，主人公の国籍が日本でありすぎたことが邪魔だった。

香港映画の「シティハンター」は，無国籍映画としての完成度でいえば「孔雀王」よりずっと上だろう。完成度とは別の次元で，香港映画は未知数の可能性をいろいろと秘めている。

illustration：Haruhisa Yamada

第131部　シューティングゲームコアシステム作成法(2)

●シューティングゲーム2

今月号のシューティングゲームコアシステム作成法では，キャラクタ表示ルーチンの制作，プログラミングガイドラインの解説など，より話がシステムらしいことに進んでいきます。

といっても，リストを見てもらえればわかるとおり，それぞれのサブルーチンはそれほど複雑なものではありません。呼び出し方法もただ単にレジスタに引数を設定してコールするだけですし，個々のルーチンは細分化された単純なものばかりです。こういったシステムを制作するうえで大切なのは，それぞれのサブルーチンをどう有機的に結びつけていくか，ということです。

今回制作したものは，キャラクタ表示関係だけですが，これだけを見てもできることが細分化され，まとめるところはまとめあげている様子がうかがえるでしょう。さらにいえば，ゲームのためのシステムということで，エラーチェックを省くなどかなり割り切って制作していることもわかると思います。

さて，こういったシステムには，必ず制約事項，プログラミングガイドラインがシステム制作者によって設定されることになります。今回もこと細かに書かれていますね。

このガイドラインというのは，アプリケーションを制作する立場の人間から見ればかなりうっとうしいものです。しかし，てんでばらばらにシステムを使っていたのでは，最大の利点である資源の共有化もままならず，システムの意味すらなくなってしまいます。もしも，どうしてもがまんができず，こういったことが窮屈に感じたならば，今度はあなた自身でよりよいものを目指していきましょう。

はっきりいって，システムはユーザーの心をつかみ，そして使わせることができれば自然と発展していくものです。せっかく作ったものが，誰にも使われずただ埋もれてしまうのはもったいないことですから。

ぜひ，ユーザーを惹きつける魅力のあるものを作るため，皆さんも頭を悩ませてみてください。

●S-OSの系譜（43）

1990年5月号では，インタプリタ言語「STACK」が登場しました。作者は，TTI，TTCシリーズでお馴染みの平井氏です。

インタプリタ言語「STACK」は，それまで発表されたコンパクトでBASICライクなTT?シリーズと，ちょっと趣を変えたスタック型言語です。代表的なスタック型言語として「FORTH」という言語がありますが，この「STACK」はスタック型BASICともいえる命令体系をもつという，一風変わった一面がありました。

また，インタプリタながらセミコンパイル機能も内蔵していて，速度的にも高速化を図れるようになっています。

さて，このスタック型言語の特徴として，計算式の逆ポーランド記法があります。前述のとおり，スタック型言語では命令に与えるパラメータがスタックに格納されます。

これは，計算式で扱うパラメータでも同様で，たとえば，

　100＋20×5

という式は，

　20　5　×　100　＋

のように記述されることになるのです。処理の手順としては，最初に20と5の値が順次スタックに積まれます。次に“×”の演算子がきた段階で，インタプリタは先ほどスタックに積まれた2つの値を取り出して演算をします。そして，その結果である100が，またスタックに積まれます。あとは同じようにパラメータがきたらスタックに積み，演算子が現れたら2つの値を取り出して演算結果をスタックに積むのです。

このような方式を利用する利点として，インタプリタ本体が式の評価を行う必要がなく，処理系が比較的簡単に作成できるのです。

しかし，逆にいえばあらかじめ式の評価を行わなくてはならない，という負担をユーザーが背負うためとっつきにくく，さらにリストを読むためにも頭の中にスタックの内容をいちいち覚えていなくてはならない，ということもあり，リストの解読も慣れるまでやりにくい，ともいわれたこともありました。それでも好きな人はとことん好きになっていくという，不思議な魅力をもった言語でした。

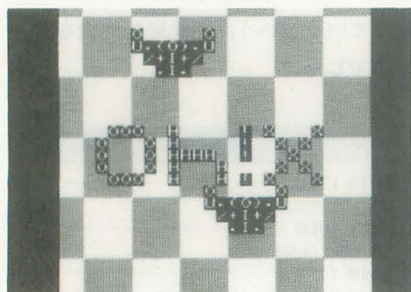
1993■インデックス

全機種共通 S-OS"SWORD"要

シューティングゲーム

コアシステム作成法(2)

Sakamaki Katumi
坂巻 克巳

今回は，システムの構成をより具体的にしたリストを掲載します。内容は，前回の画面制御関係のサブルーチンの組み込みと，キャラクター表示関係のサービスコールの制作を行います。

システム構成

先月号では，画面制御関係の仕様とそのサブルーチンを制作しました。ここでもう一度，その仕様を確認してみます。

・仮想画面は30×29キャラクタで，実画面表示エリアは26×25キャラクタである
・仮想画面は4ページ分用意され，
　VRAM3　　背景ページ
　VRAM1,2　キャラクタ表示用ページ
　VRAM0　　仮想画面合成用差分ページ
のように構成されている。VRAM3は背景画面ということになっているが，キャラクタ表示用の画面として使用することもできる。なお，VRAM0はシステム側のワークエリアであるため，ユーザーが直接アクセスすることは許されない

そして，試作したサブルーチンは，
・VRAMINIT
　仮想画面（VRAM0～3）の初期化
・BGSCROLL
　背景(VRAM3)の1ラインスクロール
・PAGEMIX
　仮想画面（VRAM1～3）を合成して，実画面エリアに表示する
以上の3つでした。前回のリストは，これらのサブルーチンがテストプログラムと一緒に組み込まれているサンプルプログラムでした。実際にシステムを作成するためには，これらのサブルーチンをまとめ上げ，コール方法などの取り決めをしっかり行う必要があります。

といってもそれほど複雑なことをするわけではありません。具体的には，サブルーチンのエントリアドレスの整理やワークエリアの整理ぐらいです。たとえば，S-OSでは，サービスルーチンのエントリアドレスをジャンプテーブルにまとめ，アプリケーションはレジスタに必要なパラメータを設定し，サービスルーチンをコールしています。

また，グラフィックパッケージのMAGICでは，それぞれのサービスコールをコマンド化してあります。アプリケーションは，それらのコマンドをデータ列にして，MAGICにそのデータ列のポインタを渡してやることになります。MAGICは順次それらのコマンドを実行してくれます。

今回作成しようとしているのは，ゲームコアシステムです。MAGICのように専用インタプリタを使って処理するのもわかりやすく，かっこいい（?）のですが，あまり重たいことはやりたくありません（効率的にやればそれほど重くないはず）。ここは，簡単にエントリアドレスとワークエリアをまとめる程度にします。

具体的に話を進めていくことにしましょう。まず，アドレスマップは図1のようになります。3000_H～$30FF_H$までが，サービスコールルーチンのエントリアドレスをまとめたテーブルと，ユーザーが使用できるコアシステムのワークエリアです。

そして，3100_H～$3FFF_H$までコアシステムの本体が入ることになります。あとは，キャラクター管理のためのワークエリア，仮想画面のワークエリアが続いています。

このメモリマップを見てもわかるとおり，一応，システムエリアとして4Kバイト取っています。これは，システム自体が完成していないためで，実際にはもっとコンパクトに収まるでしょう。まあ，とりあえず段階的に制作していくという手順を踏んでいるため，現段階でそれぞれのアドレスをきっちり確定してしまうと，システムを拡張する場合にかなりの苦労を強いられることになります。要するに完成してないんだから余裕をもってプログラムを制作しましょ，ということです。

キャラクター管理

それでは，今月のメインであるキャラクター管理ルーチンの仕様を考えていきます。

必要となるサブルーチンは，
・キャラクターのプット
・キャラクターの消去
・キャラクターワークの制御
というものがあります。技術的にはそれほど難しいものではありませんが，効率よくキャラクターを管理するためのデータ構造を考えなければなりません。たとえば，図2のように，それぞれのキャラクターが対応するレジスタの内容を書き換えることでキャラクターを管理する，という方法を取らなくてはならないのです。

しかし，どのようなキャラクターを表示するかを明確にしないとデータ構造も決まりません。今度は，キャラクター自身について考えましょう。通常，S-OSでキャラクターといえばテキスト1文字を意味しますが，ゲームなどでキャラクターといえば，動かされているキャラクターのブロック単位を意味することになります。

そのため，単一文字ごとにキャラクターを管理していたのでは，かなりの無駄を生じることになるのが明白ですね。このあたりを念頭において仕様を考えます。

・キャラクターは，1×1～N×Nキャラクタの大きさまでを扱えるようにする
・使用頻度の高いと思われる1×1～3×3キャラクタまでの専用プットルーチンと，任意の大きさを表示できるN×Nキャラクタのプットルーチンを用意する
・キャラクターは仮想画面にプットされ，実表示画面には表示しない
・キャラクターはコアシステムで一括管理し，表示，消去もシステムでサポートする

以上のことをもとにキャラクター1個あたりに必要となるワークは，表1のようなものになります。そして，システムが管理するキャラクターの個数は128個とします。1キャラクターあたり8バイトのワークが必要ですから，全部で1024バイトのワークをキャラクター管理ワークとして使用することになります。

利用方法

今回制作したサービスコールは表2にまとめておきました。あとは，サンプルプログラム（リスト1）を簡単に解説していきます。

処理の流れとしては，
・キャラクター管理ワークエリアの初期化（VSPREGINIT）
・仮想画面の初期化（VRAMINIT）
・キャラクターの登録（VSPREGSET）
以上が初期化ルーチンで，メインルーチンでは，
・背景のセット（VRAM3を直接アクセス）
・背景のスクロール（BGSCROLL）
・キャラクターの移動（VSPREGWORKをVSPREGSETで書き換えている）
・キャラクターを仮想画面に書き込む（VSPPUT）
・仮想画面の表示（PAGEMIX）
・表示したキャラクターを仮想画面から消去する（VSPERASE）
のようになっています。

最初にキャラクターの登録をする。そして，その登録したキャラクターを仮想画面にセットしてから，実表示画面に仮想画面の内容を表示する。最後には仮想画面にセットしたキャラクターを消去する。というような手順が基本的なものとなります。

ガイドライン

最後に，今回制作したリスト2のコアシステムを使ううえで，守らなければならない約束事，プログラミングガイドラインを書き連ねていきます。
・サービスコールは公表されたもの以外，使用してはならない

つまり，システム内部で利用されているサブルーチンや，エントリアドレスを無視したサービスコールをしてはならない，ということです。また，ワークも公表されているもの以外，いじることは許されません。これは，S-OSも同様で将来的に拡張が行われた場合でも，古いアプリケーションを動かせるようにするためです。
・仮想画面は，必ずVRAM3,2,1,0の順番で1024バイト単位を確保しておかなければならない

つまり，仮想画面のワークエリアは，常に4Kバイトを確保しておかなくてはならない，ということです。とりあえず，仮想画面が格納されている先頭アドレスを自由に動かせますが，確保したワークエリアが4Kバイトに満たない場合，アプリケーションの動作保証はありません。
・VRAM0を直接いじることは許されない

これは何度も説明があるとおり，システムコールPAGEMIXで，VRAM0は差分画面として使っています。ですから，下手に書き換えてしまうと表示が正常に行われなくなってしまうのです。VRAM1～3とVSPREGWORKについては，ユーザーが自由にアクセスしてもかまいません。
・サービスコールは，パラメータに対するエラーチェックを行っていない

パラメータチェックはすべてアプリケーション側で行う，という前提でコアシステムの高速化を目指してプログラミングしたためです。

だいだい，以上のようなことを守ってアプリケーションを制作するようにしてください。また，サービスコールが追加されていくと，いろいろ制限が出てくると思いますが，そのときは順次解説していくことにします。

＊　＊　＊

キャラクタの表示部分については，以上のようにまとまりました。面倒な部分もありますが，正しく楽しく使いましょう。来月は，もう少しこのキャラクター表示部分をつつくか，もしくはよりゲームに密着したサービスコールを制作していくつもりです。うん，がんばらなくちゃ。

図1　アドレスマップ

図2　レジスタの役割

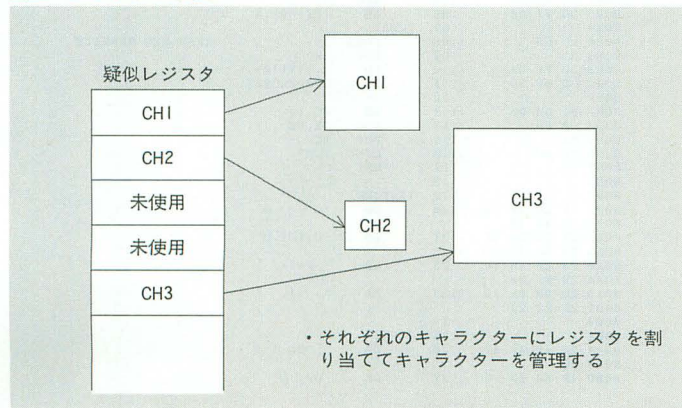



表1　疑似スプライトレジスタ（1スプライト＝8バイト）

VSPREGWORK	
オフセット	機　　能
00	FLAG　00：NO USE 01：1×1キャラクタ 02：2×2キャラクタ 03：3×3キャラクタ 04：N×Nキャラクタ
01	ページ番号（1～3）
02	X,Y座標
04	PCGADDR(FLAG=1のときキャラクタコード)
06	キャラクターX,Yサイズ(FLAG=4のときのみ使用)

・使用できる疑似スプライトワークは128個
・ワークは1024バイト確保しなくてはならない

表2　疑似スプライトルーチン&ワーク詳細

ラベル	機能	破壊レジスタ
ADDRCAL (3100H)	入力されたX,Y座標とページ番号に従ってアドレス計算をする IN　DE=X,Y座標 A＝ページ番号 OUT HL=アドレス	AF,HL,DE,BC
CHR1PUT (3103H)	単一キャラクタープット IN　DE=表示Y,X座標 B＝キャラクター番号 A＝ページ番号 OUT　なし	AF,HL,DE,BC
CHR2PUT (3103H)	2×2キャラクタープット IN　DE=表示Y,X座標 HL=PCGアドレス A＝ページ番号 OUT　なし	AF,HL,DE,BC
CHR3PUT (3106H)	3×3キャラクタープット IN　DE=表示Y,X座標 HL=PCGアドレス A＝ページ番号 OUT　なし	AF,HL,DE,BC
CHRNPUT (3109H)	N×Nキャラクタープット IN　DE=表示Y,X座標 HL=PCGアドレス BC=X,Yキャラクターサイズ A＝ページ番号 OUT　なし	AF,HL,DE,BC
VSPREGSET (310CH)	疑似スプライトレジスタへの書き込み IN　DE=データ列ポインタアドレス A＝疑似スプライト番号 ※データ列詳細は表1を参照のこと	AF,HL,DE,BC
VSPREGSET2 (310FH)	簡易疑似スプライトレジスタへの書き込み IN　A＝スプライト番号 C＝パラメータオフセット E＝パラメータ	AF,HL,DE,BC
VSPREGERA (3112H)	疑似スプライトキャラクターの消去 IN　A＝スプライト番号	AF,HL,DE
VSPPUT (3115H)	疑似スプライトレジスタワークに定義されているキャラクタをプット IN,OUT　なし	AF,B,IX AF',HL',DE' BC'
VSPERASE (3118H)	疑似スプライトレジスタワークに定義されているキャラクターを消去 IN,OUT　なし	AF,B,IX AF',HL',DE' BC'
BGSCROLL (311EH)	背景（VRAM3）の1ラインスクロール IN,OUT　なし	AF,HL,DE,BC
PAGEMIX (3121H)	VRAM0～3を合成して画面に表示 IN,OUT　なし	AF,HL,DE,BC B'
VRAMINIT (3124H)	仮想画面の初期化 IN,OUT　なし	HL,DE,BC
VSPREGINIT (3127H)	疑似スプライトレジスタワークの初期化 IN,OUT　なし	HL,DE,BC
VRAMADDR (302AH)	仮想画面VRAM3の先頭アドレス	
VSPREGADDR (302CH)	疑似スプライトレジスタワークエリアアドレス	

リスト1

```
0000                 1
0000                 2  ;*************************
0000                 3  ;  SAMPLE PROGRAM
0000                 4  ;*************************
0000                 5
1FF4 P               6  #PRINT  EQU .   $1FF4
201E P               7  #LOC    EQU     $201E
1FD0 P               8  #GETKEY EQU     $1FD0
0000                 9
3000 P              10  @ADDRCAL  EQU   $3000
3003 P              11  @CHR1PUT  EQU   $3003
3006 P              12  @CHR2PUT  EQU   $3006
3009 P              13  @CHR3PUT 'EQU   $3009
300C P              14  @CHRNPUT  EQU   $300C
300F P              15  @VSPREGSET EQU  $300F
3012 P              16  @VSPREGSET2 EQU $3012
3015 P              17  @VSPREGERA EQU  $3015
3018 P              18  @VSPPUT   EQU   $3018
301B P              19  @VSPERASE EQU   $301B
301E P              20  @BGSCROLL EQU   $301E
3021 P              21  @PAGEMIX  EQU   $3021
3024 P              22  @VRAMINIT EQU   $3024
3027 P              23  @VSPREGINIT EQU $3027
0000                24
302A P              25  VRAMADDR  EQU   $302A
0000                26
0000                27
8000                28          ORG     $8000
8000                29
8000                30  TEST:
8000 3E 0C          31      LD    A,$0C
8002 CD F4 1F       32      CALL  #PRINT
8005 CD 27 30       33      CALL  @VSPREGINIT
8008 CD 24 30       34      CALL  @VRAMINIT
800B                35
800B 11 04 81       36      LD    DE,TESTCHRWORK
800E 3E 00          37      LD    A,00
8010 06 06          38      LD    B,06
8012                39  TT2:
8012 C5             40      PUSH  BC
8013 D5             41      PUSH  DE
8014 F5             42      PUSH  AF
8015 CD 0F 30       43      CALL  @VSPREGSET
8018 F1             44      POP   AF
8019 3C             45      INC   A
801A D1             46      POP   DE
801B 21 0B 00       47      LD    HL,11
801E 19             48      ADD   HL,DE
801F EB             49      EX    DE,HL
8020 C1             50      POP   BC
8021 10 EF          51      DJNZ  TT2
8023                52  TEST2:
8023 CD 46 81       53      CALL  BGTEST
8026 CD 1E 30       54      CALL  @BGSCROLL
8029                55
8029 CD 3D 80       56      CALL  TESTCHRMAIN
802C                57
802C CD 18 30       58      CALL  @VSPPUT
802F CD 21 30       59      CALL  @PAGEMIX
8032 CD 1B 30       60      CALL  @VSPERASE
8035                61
8035 CD D0 1F       62      CALL  #GETKEY
8038 FE 20          63      CP    $20
803A 20 E7          64      JR    NZ,TEST2
803C C9             65      RET
803D                66
803D                67  TESTCHRMAIN:
803D DD 21 04 81    68      LD    IX,TESTCHRWORK
8041 06 06          69      LD    B,06
8043                70  TCM2:
8043 C5             71      PUSH  BC
8044 DD 7E 02       72      LD    A,(IX+02)      ;MOVE
8047 DD 86 08       73      ADD   A,(IX+08)
804A DD 77 02       74      LD    (IX+02),A
804D                75
804D DD 7E 03       76      LD    A,(IX+03)
8050 DD 86 09       77      ADD   A,(IX+09)
8053 DD 77 03       78      LD    (IX+03),A
8056                79
8056 DD 7E 02       80      LD    A,(IX+02)      ;X CHECK
8059 FE 02          81      CP    02
805B 38 04          82      JR    C,TCMX2
805D FE 16          83      CP    22
805F 38 0D          84      JR    C,TCMY
8061                85  TCMX2:
8061 DD 46 08       86      LD    B,(IX+08)      ;IDOURYOU GET
8064 90             87      SUB   B
8065 DD 77 02       88      LD    (IX+02),A
8068                89
8068 78             90      LD    A,B
8069 ED 44          91      NEG
806B DD 77 08       92      LD    (IX+08),A
806E                93  TCMY:
806E DD 7E 03       94      LD    A,(IX+03)      ;Y CHECK
8071 FE 02          95      CP    02
8073 38 04          96      JR    C,TCMY2
8075 FE 16          97      CP    22
8077 38 0D          98      JR    C,TCM3
8079                99  TCMY2:
8079 DD 46 09      100      LD    B,(IX+09)      ;IDOURYOU GET
807C 90            101      SUB   B
807D DD 77 03      102      LD    (IX+03),A
8080               103
8080 78            104      LD    A,B
8081 ED 44         105      NEG
8083 DD 77 09      106      LD    (IX+09),A
8086               107  TCM3:
8086 DD E5         108      PUSH  IX             ;VSP REG REWRITE
8088 D1            109      POP   DE
8089 DD 7E 0A      110      LD    A,(IX+10)
808C CD 0F 30      111      CALL  @VSPREGSET
808F               112
808F 01 0B 00      113      LD    BC,11
8092 DD 09         114      ADD   IX,BC
8094 C1            115      POP   BC
8095 10 AC         116      DJNZ  TCM2
8097 C9            117      RET
8098               118
8098               119  TESTCHR:
8098 4F 20 20 20   120      DB    'O     O'
809C 20 20 4F
809F 55 2E 28 4F   121      DB    'U.(O).U'
80A3 29 2E 55
80A6 20 2F 2B 49   122      DB    ' /+I+/ '
80AA 2B 2F 20
80AD 20 20 2E 49   123      DB    '  .I.  '
80B1 2E 20 20
80B4               124
80B4               125  OCHR:
80B4 20 4F 4F 4F   126      DB    ' OOO '
80B8 20
80B9 4F 20 20 20   127      DB    'O   O'
```

▶X68030を見て思わず欲しいと感じた。けど，よく考えてみるとなんに使うつもりなんだろう。いまやっていることといえば，CG，音楽の観賞，ゲーム，通信，C，アセンブラの入門……う～ん別に必要ないかな。　外山　新一(20)愛知県

```
80BD 4F
80BE 4F 20 20 20     128         DB     'O   O'
80C2 4F
80C3 4F 20 20 20     129         DB     'O   O'
80C7 4F
80C8 20 4F 4F 4F     130         DB     ' OOO '
80CC 20
80CD                 131    HCHR:
80CD 48 20 20 20     132         DB     'H    '
80D1 20
80D2 48 20 20 20     133         DB     'H    '
80D6 20
80D7 48 48 48 48     134         DB     'HHHH '
80DB 20
80DC 48 20 20 20     135         DB     'H   H'
80E0 48
80E1 48 20 20 20     136         DB     'H   H'
80E5 48
80E6                 137    ICHR:
80E6 21              138         DB     '!'
80E7 21              139         DB     '!'
80E8 21              140         DB     '!'
80E9 20              141         DB     ' '
80EA 21              142         DB     '!'
80EB                 143    XCHR:
80EB 58 20 20 20     144         DB     'X   X'
80EF 58
80F0 20 58 20 58     145         DB     ' X X '
80F4 20
80F5 20 20 58 20     146         DB     '  X  '
80F9 20
80FA 20 58 20 58     147         DB     ' X X '
80FE 20
80FF 58 20 20 20     148         DB     'X   X'
8103 58
8104                 149
8104                 150    ; TEST CHARACTER WORK AREA
8104                 151
8104                 152    TESTCHRWORK:
8104 04              153         DB     04              ;FLAG
8105 01              154         DB     01              ;PUT PAGE
8106 05 0C           155         DB     05,12           ;DISP XY
8108 B4 80           156         DW     OCHR            ;PCG ADDRESS
810A 05 05           157         DB     05,05           ;CHR SIZE
810C 00 00           158         DB     0,0             ;XY IDOURYOU
810E 00              159         DB     00              ;VSP NUM
810F                 160
810F 04              161         DB     04
8110 01              162         DB     01
8111 0B 0C           163         DB     11,12
8113 CD 80           164         DW     HCHR
8115 05 05           165         DB     05,05
8117 00 00           166         DB     0,0
8119 01              167         DB     01
811A                 168
811A 04              169         DB     04
811B 01              170         DB     01
811C 11 0C           171         DB     17,12
811E E6 80           172         DW     ICHR
8120 01 05           173         DB     01,05
8122 00 00           174         DB     0,0
8124 02              175         DB     02
8125                 176
8125 04              177         DB     04
8126 01              178         DB     01
8127 13 0C           179         DB     19,12
8129 EB 80           180         DW     XCHR
812B 05 05           181         DB     05,05
812D 00 00           182         DB     00,00
812F 03              183         DB     03
8130                 184
8130 04              185         DB     04
8131 02              186         DB     02
8132 0F 12           187         DB     15,18
8134 98 80           188         DW     TESTCHR
8136 07 04           189         DB     07,04
8138 FF FF           190         DB     -1,-1
813A 04              191         DB     04
813B                 192
813B 04              193         DB     04
813C 02              194         DB     02
813D 07 03           195         DB     07,03
813F 98 80           196         DW     TESTCHR
8141 07 04           197         DB     07,04
8143 01 01           198         DB     1,1
8145 05              199         DB     05
8146                 200
8146                 201    BGTEST:
8146 3A 73 81        202         LD     A,(BGCNT)
8149 3D              203         DEC    A
814A 20 17           204         JR     NZ,BGT2
814C 21 78 81        205         LD     HL,BGDATA1
814F 3A 76 81        206         LD     A,(BGFLAG)
8152 FE 00           207         CP     0
8154 28 03           208         JR     Z,BGT3
8156 21 96 81        209         LD     HL,BGDATA2
8159                 210    BGT3:
8159 EE 01           211         XOR    1
815B 32 76 81        212         LD     (BGFLAG),A
815E 22 74 81        213         LD     (BGADDR),HL
8161 3E 05           214         LD     A,05
8163                 215    BGT2:
8163 32 73 81        216         LD     (BGCNT),A
8166 2A 74 81        217         LD     HL,(BGADDR)
8169 ED 5B 2A 30     218         LD     DE,(VRAMADDR)
816D 01 1E 00        219         LD     BC,30
8170 ED B0           220         LDIR
8172 C9              221         RET
8173                 222
8173                 223    ; WORK AREA
8173                 224
8173                 225    BGCNT:
8173 05              226         DB     05
8174                 227    BGADDR:
8174 78 81           228         DW     BGDATA1
8176                 229    BGFLAG:
8176 00              230         DB     00
8177                 231
8177                 232    VSPCNT:                         ;VSPRITE COUNTER
8177 00              233         DB     00
8178                 234
8178                 235
8178                 236    ; BG TEST DATA
8178                 237
8178                 238    BGDATA1:
8178 7B 7B 7B 7B     239         DB     $7B,$7B,$7B,$7B,$7D,$7D,$7D,$7D
817C 7D 7D 7D 7D
8180 7B 7B 7B 7B     240         DB     $7B,$7B,$7B,$7B,$7D,$7D,$7D,$7D
8184 7D 7D 7D 7D
8188 7B 7B 7B 7B     241         DB     $7B,$7B,$7B,$7B,$7D,$7D,$7D,$7D
818C 7D 7D 7D 7D
8190 7B 7B 7B 7B     242         DB     $7B,$7B,$7B,$7B,$7D,$7D
8194 7D 7D
8196                 243
8196                 244    BGDATA2:
8196 7D 7D 7D 7D     245         DB     $7D,$7D,$7D,$7D,$7B,$7B,$7B,$7B
819A 7B 7B 7B 7B
819E 7D 7D 7D 7D     246         DB     $7D,$7D,$7D,$7D,$7B,$7B,$7B,$7B
81A2 7B 7B 7B 7B
81A6 7D 7D 7D 7D     247         DB     $7D,$7D,$7D,$7D,$7B,$7B,$7B,$7B
81AA 7B 7B 7B 7B
81AE 7D 7D 7D 7D     248         DB     $7D,$7D,$7D,$7D,$7B,$7B
81B2 7B 7B
81B4                 249
81B4                 250
```

リスト2

```
0000                  1
0000                  2
0000                  3   ;*********************************
0000                  4   ; SHOOTING  GAME
0000                  5   ;          CORE SYSTEM
0000                  6   ;*********************************
0000                  7
0000                  8            OFFSET  $6000
3000                  9            ORG     $3000
3000                 10
1FF4 P               11   #PRINT  EQU      $1FF4
201E P               12   #LOC    EQU      $201E
3000                 13
3000                 14   @ADDRCAL
3000 C3 00 31        15       JP    ADDRCAL
3003                 16   @CHR1PUT
3003 C3 1A 31        17       JP    CHR1PUT
3006                 18   @CHR2PUT
3006 C3 21 31        19       JP    CHR2PUT
3009                 20   @CHR3PUT
3009 C3 38 31        21       JP    CHR3PUT
300C                 22   @CHRNPUT
300C C3 54 31        23       JP    CHRNPUT
300F                 24   @VSPREGSET
300F C3 93 31        25       JP    VSPREGSET
3012                 26   @VSPREGSET2
3012 C3 A5 31        27       JP    VSPREGSET2
3015                 28   @VSPREGERA
3015 C3 B5 31        29       JP    VSPREGERA
3018                 30   @VSPPUT
3018 C3 C3 31        31       JP    VSPPUT
301B                 32   @VSPERASE
301B C3 11 32        33       JP    VSPERASE
301E                 34   @BGSCROLL
301E C3 5D 32        35       JP    BGSCROLL
3021                 36   @PAGEMIX
3021 C3 81 32        37       JP    PAGEMIX
3024                 38   @VRAMINIT
3024 C3 CA 32        39       JP    VRAMINIT
3027                 40   @VSPREGINIT
3027 C3 D8 32        41       JP    VSPREGINIT
302A                 42
302A                 43   ; VRAM ADDRESS WORK
302A                 44
302A                 45   VRAMADDR:
302A 00 44           46       DW    VRAM3
302C                 47
302C                 48   ; VSPRITE REG ADDR WORK
302C                 49
302C                 50   VSPREGADDR:
302C 00 40           51       DW    VSPREGWORK
302E                 52
302E                 53   ; VSPTITE ERASE CHR
302E                 54
302E                 55   VSPERACHR:
302E 20 20 20 20     56       DB    '         '
3032 20 20 20 20
3036 20
3037                 57
3100                 58       ORG      $3100
3100                 59
3100                 60   ; VRAM PUT ADDR
3100                 61   ; IN DE=DISP XY
3100                 62   ;     A=PAGE NUMBER
3100                 63   ; OUT HL=ADDRESS
3100                 64
3100                 65   ADDRCAL:
3100 2A 2A 30        66       LD    HL,(VRAMADDR)
3103                 67
3103 47              68       LD    B,A              ;PAGE ADDR CALC
3104 3E 03           69       LD    A,03
3106 90              70       SUB   B
3107 87              71       ADD   A,A
3108 87              72       ADD   A,A
3109 84              73       ADD   A,H
310A 67              74       LD    H,A
310B                 75
310B 4B              76       LD    C,E              ;ADD X
310C 06 00           77       LD    B,00
310E 09              78       ADD   HL,BC
310F                 79
310F E5              80       PUSH  HL
3110 7A              81       LD    A,D              ;Y*30
3111 11 1E 00        82       LD    DE,30
3114 CD 86 31        83       CALL  MUL8
3117 D1              84       POP   DE
3118 19              85       ADD   HL,DE
3119 C9              86       RET
311A                 87
311A                 88   ; 1*1 CHARACTER PUT
311A                 89   ;  IN DE=PUT XY
311A                 90   ;     B=PUT CHARACTER CODE
```

▶仕事の帰りにOh!Xを買った。裏表紙を見て驚いた。思わず，電車の中で見せびらかすように読んでしまった。　　伴　武士(22)千葉県

```
311A                    91  ;     A=PAGE NUMBER
311A                    92
311A                    93  CHR1PUT:
311A C5                 94       PUSH  BC
311B CD 00 31           95       CALL  ADDRCAL          ;PUT ADDR CALC
311E C1                 96       POP   BC
311F 70                 97       LD    (HL),B
3120 C9                 98       RET
3121                    99
3121                   100  ; 2*2 CHARACTER PUT
3121                   101  ;  IN DE=PUT XY
3121                   102  ;     HL=PCG ADDR
3121                   103  ;     A=PAGE NUMBER
3121                   104
3121                   105  CHR2PUT:
3121 E5                106       PUSH  HL
3122 CD 00 31          107       CALL  ADDRCAL          ;PUT ADDR CALC
3125 D1                108       POP   DE
3126                   109
3126 01 1D 00          110       LD    BC,29
3129 1A                111       LD    A,(DE)
312A 77                112       LD    (HL),A
312B 13                113       INC   DE
312C 23                114       INC   HL
312D 1A                115       LD    A,(DE)
312E 77                116       LD    (HL),A
312F 13                117       INC   DE
3130                   118
3130 09                119       ADD   HL,BC
3131 1A                120       LD    A,(DE)
3132 77                121       LD    (HL),A
3133 23                122       INC   HL
3134 13                123       INC   DE
3135 1A                124       LD    A,(DE)
3136 77                125       LD    (HL),A
3137 C9                126       RET
3138                   127
3138                   128  ; 3*3 CHARACTER PUT
3138                   129  ;  IN DE=PUT XY
3138                   130  ;     HL=PCG ADDR
3138                   131  ;     A=PAGE NUMBER
3138                   132
3138                   133  CHR3PUT:
3138 E5                134       PUSH  HL
3139 CD 00 31          135       CALL  ADDRCAL          ;PUT ADDR CALC
313C D1                136       POP   DE
313D                   137
313D 3E 03             138       LD    A,03
313F                   139  CH3P2:
313F 08                140       EX    AF,AF'
3140 01 1C 00          141       LD    BC,28
3143 1A                142       LD    A,(DE)
3144 77                143       LD    (HL),A
3145 23                144       INC   HL
3146 13                145       INC   DE
3147 1A                146       LD    A,(DE)
3148 77                147       LD    (HL),A
3149 23                148       INC   HL
314A 13                149       INC   DE
314B 1A                150       LD    A,(DE)
314C 77                151       LD    (HL),A
314D 13                152       INC   DE
314E 09                153       ADD   HL,BC
314F 08                154       EX    AF,AF'
3150 3D                155       DEC   A
3151 20 EC             156       JR    NZ,CH3P2
3153 C9                157       RET
3154                   158
3154                   159  ; N*N CHARACTER PUT
3154                   160  ;  IN DE=PUT XY
3154                   161  ;     HL=PCG ADDR
3154                   162  ;     BC=CHR SIZE XY
3154                   163  ;     A=PAGE NUMBER
3154                   164
3154                   165  CHRNPUT:
3154 C5                166       PUSH  BC
3155 E5                167       PUSH  HL
3156 CD 00 31          168       CALL  ADDRCAL          ;PUT ADDR CALC
3159 D1                169       POP   DE
315A C1                170       POP   BC
315B                   171  CHNP2:
315B C5                172       PUSH  BC
315C E5                173       PUSH  HL
315D                   174  CHNP3:
315D 1A                175       LD    A,(DE)
315E 77                176       LD    (HL),A
315F 13                177       INC   DE
3160 23                178       INC   HL
3161 0D                179       DEC   C
3162 20 F9             180       JR    NZ,CHNP3
3164                   181
3164 E1                182       POP   HL
3165 01 1E 00          183       LD    BC,30
3168 09                184       ADD   HL,BC            ;NEXT LINE
3169 C1                185       POP   BC
316A 10 EF             186       DJNZ  CHNP2
316C C9                187       RET
316D                   188
316D                   189  ; N*N CHARACTER ERASE
316D                   190
316D                   191  CHRNERASE:
316D C5                192       PUSH  BC
316E E5                193       PUSH  HL
316F CD 00 31          194       CALL  ADDRCAL
3172 D1                195       POP   DE
3173 C1                196       POP   BC
3174                   197  CHER2:
3174 C5                198       PUSH  BC
3175 E5                199       PUSH  HL
3176                   200  CHER3:
3176 36 20             201       LD    (HL),$20
3178 13                202       INC   DE
3179 23                203       INC   HL
317A 0D                204       DEC   C
317B 20 F9             205       JR    NZ,CHER3
317D                   206
317D E1                207       POP   HL
317E 01 1E 00          208       LD    BC,30
3181 09                209       ADD   HL,BC
3182 C1                210       POP   BC
3183 10 EF             211       DJNZ  CHER2
3185 C9                212       RET
3186                   213
3186                   214  ; 16bit*8bit
```

```
3186                   215  ;  HL=DE*A
3186                   216
3186                   217  MUL8:
3186 21 00 00          218       LD    HL,0000
3189 06 08             219       LD    B,08
318B                   220  ML83:
318B 29                221       ADD   HL,HL
318C 87                222       ADD   A,A
318D 30 01             223       JR    NC,ML82
318F 19                224       ADD   HL,DE
3190                   225  ML82:
3190 10 F9             226       DJNZ  ML83
3192 C9                227       RET
3193                   228
3193                   229  ; VSPRITE REG DATA SET
3193                   230  ;  IN DE=DATA POINTER ADDRESS
3193                   231  ;     A=USE VSP NUMBER
3193                   232  ;    00:FLAG(CHR NUMBER)
3193                   233  ;    01:PAGE NUMBER
3193                   234  ;    02:DISP XY
3193                   235  ;    04:PCG ADDRESS
3193                   236  ;    06:CHARACTER XY SIZE
3193                   237
3193                   238  VSPREGSET:
3193 ED 4B 2C 30       239       LD    BC,(VSPREGADDR)
3197 6F                240       LD    L,A
3198 26 00             241       LD    H,00
319A 29                242       ADD   HL,HL
319B 29                243       ADD   HL,HL
319C 29                244       ADD   HL,HL
319D 09                245       ADD   HL,BC
319E 01 08 00          246       LD    BC,08
31A1 EB                247       EX    DE,HL
31A2 ED B0             248       LDIR
31A4 C9                249       RET
31A5                   250
31A5                   251  ; VSPRITE REG EASY DATA SET
31A5                   252  ;  IN A=VSPRITE NUMBER
31A5                   253  ;     C=PARAMETER OFFSET
31A5                   254  ;     E=PARAMETER
31A5                   255
31A5                   256  VSPREGSET2:
31A5 ED 5B 2C 30       257       LD    DE,(VSPREGADDR)
31A9 6F                258       LD    L,A
31AA 26 00             259       LD    H,00
31AC 06 00             260       LD    B,00
31AE 29                261       ADD   HL,HL
31AF 29                262       ADD   HL,HL
31B0 29                263       ADD   HL,HL
31B1 19                264       ADD   HL,DE
31B2 09                265       ADD   HL,BC
31B3 73                266       LD    (HL),E
31B4 C9                267       RET
31B5                   268
31B5                   269  ; VSPRITE FLAG ERASE
31B5                   270  ;  IN A=VSPRITE NUMBER
31B5                   271
31B5                   272  VSPREGERA:
31B5 ED 5B 2C 30       273       LD    DE,(VSPREGADDR)
31B9 6F                274       LD    L,A
31BA 26 00             275       LD    H,00
31BC 29                276       ADD   HL,HL
31BD 29                277       ADD   HL,HL
31BE 29                278       ADD   HL,HL
31BF 19                279       ADD   HL,DE
31C0 36 00             280       LD    (HL),00          ;FLAG ERASE
31C2 C9                281       RET
31C3                   282
31C3                   283  ; VSPRITE CHARACTER PUT
31C3                   284
31C3                   285  VSPPUT:
31C3 DD 2A 2C 30       286       LD    IX,(VSPREGADDR)
31C7 06 80             287       LD    B,128
31C9                   288  VSPP2:
31C9 D9                289       EXX
31CA DD 7E 00          290       LD    A,(IX+0)
31CD FE 00             291       CP    00
31CF 28 37             292       JR    Z,VSPP3
31D1                   293
31D1 08                294       EX    AF,AF'
31D2 DD 7E 01          295       LD    A,(IX+1)         ;PAGE NUMBER GET
31D5 08                296       EX    AF,AF'
31D6 DD 5E 02          297       LD    E,(IX+2)         ;DISP XY
31D9 DD 56 03          298       LD    D,(IX+3)
31DC DD 6E 04          299       LD    L,(IX+4)         ;PCG ADDRESS
31DF DD 66 05          300       LD    H,(IX+5)
31E2                   301
31E2 3D                302       DEC   A                ;FLAG=1:1*1 CHR
31E3 28 1E             303       JR    Z,VSPCH1
31E5 3D                304       DEC   A
31E6 28 15             305       JR    Z,VSPCH2
31E8 3D                306       DEC   A
31E9 28 0C             307       JR    Z,VSPCH3
31EB                   308  VSPCHN:
31EB 08                309       EX    AF,AF'
31EC DD 4E 06          310       LD    C,(IX+6)         ;CHR SIZE
31EF DD 46 07          311       LD    B,(IX+7)
31F2 CD 54 31          312       CALL  CHRNPUT
31F5 18 11             313       JR    VSPP3
31F7                   314  VSPCH3:
31F7 08                315       EX    AF,AF'
31F8 CD 38 31          316       CALL  CHR3PUT
31FB 18 0B             317       JR    VSPP3
31FD                   318  VSPCH2:
31FD 08                319       EX    AF,AF'
31FE CD 21 31          320       CALL  CHR2PUT
3201 18 05             321       JR    VSPP3
3203                   322  VSPCH1:
3203 08                323       EX    AF,AF'
3204 45                324       LD    B,L
3205 CD 1A 31          325       CALL  CHR1PUT
3208                   326  VSPP3:
3208 11 08 00          327       LD    DE,0008
320B DD 19             328       ADD   IX,DE
320D D9                329       EXX
320E 10 B9             330       DJNZ  VSPP2
3210 C9                331       RET
3211                   332
3211                   333  ; VSPRITE CHARACTER ERASE
3211                   334
3211                   335  VSPERASE:
3211 DD 2A 2C 30       336       LD    IX,(VSPREGADDR)
3215 06 80             337       LD    B,128
3217                   338  VSE2:
```

```
3217 D9            339        EXX
3218 DD 7E 00      340        LD    A,(IX+0)
321B FE 00         341        CP    00
321D 28 35         342        JR    Z,VSE3
321F               343
321F 08            344        EX    AF,AF'
3220 DD 7E 01      345        LD    A,(IX+1)       ;PAGE NUMBER GET
3223 08            346        EX    AF,AF'
3224 DD 5E 02      347        LD    E,(IX+2)       ;DISP XY
3227 DD 56 03      348        LD    D,(IX+3)
322A 21 2E 30      349        LD    HL,VSPERACHR   ;CHR ADDR
322D               350
322D 3D            351        DEC   A
322E 28 1E         352        JR    Z,VSECH1
3230 3D            353        DEC   A
3231 28 15         354        JR    Z,VSECH2
3233 3D            355        DEC   A
3234 28 0C         356        JR    Z,VSECH3
3236               357    VSECHN:
3236 DD 4E 06      358        LD    C,(IX+6)       ;CHR SIZE
3239 DD 46 07      359        LD    B,(IX+7)
323C 08            360        EX    AF,AF'
323D CD 6D 31      361        CALL  CHRNERASE
3240 18 12         362        JR    VSE3
3242               363    VSECH3:
3242 08            364        EX    AF,AF'
3243 CD 38 31      365        CALL  CHR3PUT
3246 18 0C         366        JR    VSE3
3248               367    VSECH2:
3248 08            368        EX    AF,AF'
3249 CD 21 31      369        CALL  CHR2PUT
324C 18 06         370        JR    VSE3
324E               371    VSECH1
324E 08            372        EX    AF,AF'
324F 06 20         373        LD    B,$20
3251 CD 1A 31      374        CALL  CHR1PUT
3254               375    VSE3:
3254 11 08 00      376        LD    DE,0008
3257 DD 19         377        ADD   IX,DE
3259 D9            378        EXX
325A 10 BB         379        DJNZ  VSE2
325C C9            380        RET
325D               381
325D               382    ; VRAM3(BG) 1LINE SCROLL
325D               383
325D               384    BGSCROLL:
325D 01 48 03      385        LD    BC,28*30
3260 2A 2A 30      386        LD    HL,(VRAMADDR)
3263 09            387        ADD   HL,BC
3264 5D            388        LD    E,L
3265 54            389        LD    D,H
3266 01 1E 00      390        LD    BC,30
3269 B7            391        OR    A
326A ED 42         392        SBC   HL,BC
326C               393
326C 3E 1C         394        LD    A,28
326E               395    BGS2:
326E 01 1E 00      396        LD    BC,30
3271 ED B0         397        LDIR
3273 01 3C 00      398        LD    BC,60
3276 B7            399        OR    A
3277 ED 42         400        SBC   HL,BC
3279 EB            401        EX    DE,HL
327A ED 42         402        SBC   HL,BC
327C EB            403        EX    DE,HL
327D 3D            404        DEC   A
327E 20 EE         405        JR    NZ,BGS2
3280 C9            406        RET
3281               407
3281               408    ; VRAM0-3 MIX
3281               409
3281               410    PAGEMIX:
3281 2A 2A 30      411        LD    HL,(VRAMADDR)
3284 11 3E 00      412        LD    DE,62
3287 19            413        ADD   HL,DE
3288               414
3288 16 00         415        LD    D,00           ;DISP START
328A D9            416        EXX
328B 06 19         417        LD    B,25
328D               418    PM1:
328D D9            419        EXX
328E 1E 05         420        LD    E,05
3290               421
3290 06 1A         422        LD    B,26
3292               423    PM2:
3292 4E            424        LD    C,(HL)         ;PAGE3 CHR GET
3293               425
3293 24            426        INC   H              ;NEXT PAGE
3294 24            427        INC   H
3295 24            428        INC   H
3296 24            429        INC   H
3297               430
3297 7E            431        LD    A,(HL)         ;PAGE2 CHR GET
3298 FE 20         432        CP    $20
329A 28 01         433        JR    Z,PM3
329C 4F            434        LD    C,A
329D               435    PM3:
329D 24            436        INC   H              ;NEXT PAGE
329E 24            437        INC   H
329F 24            438        INC   H
32A0 24            439        INC   H
32A1               440
32A1 7E            441        LD    A,(HL)         ;PAGE1 CHR GET
32A2 FE 20         442        CP    $20
32A4 28 01         443        JR    Z,PM4
32A6 4F            444        LD    C,A
32A7               445    PM4:
32A7 24            446        INC   H              ;NEXT PAGE
32A8 24            447        INC   H
32A9 24            448        INC   H
32AA 24            449        INC   H
32AB               450
32AB 7E            451        LD    A,(HL)         ;PAGE0 CHR GET
32AC B9            452        CP    C
32AD 28 0A         453        JR    Z,PM6
32AF               454
32AF 71            455        LD    (HL),C         ;CHR REWRITE
32B0 EB            456        EX    DE,HL
32B1 CD 1E 20      457        CALL  #LOC
32B4 EB            458        EX    DE,HL
32B5 79            459        LD    A,C
32B6 CD F4 1F      460        CALL  #PRINT
32B9               461    PM6:
32B9 7C            462        LD    A,H
32BA D6 0C         463        SUB   12
32BC 67            464        LD    H,A
32BD               465
32BD 23            466        INC   HL
32BE 1C            467        INC   E
32BF               468
32BF 10 D1         469        DJNZ  PM2
32C1               470
32C1 23            471        INC   HL
32C2 23            472        INC   HL
32C3 23            473        INC   HL
32C4 23            474        INC   HL
32C5 14            475        INC   D
32C6               476
32C6 D9            477        EXX
32C7 10 C4         478        DJNZ  PM1
32C9               479
32C9 C9            480        RET
32CA               481
32CA               482    ; VRAM WORK INIT
32CA               483
32CA               484    VRAMINIT:
32CA 2A 2A 30      485        LD    HL,(VRAMADDR)
32CD 5D            486        LD    E,L
32CE 54            487        LD    D,H
32CF 13            488        INC   DE
32D0 36 20         489        LD    (HL),$20
32D2 01 FF 0F      490        LD    BC,1024*4-1
32D5 ED B0         491        LDIR
32D7 C9            492        RET
32D8               493
32D8               494    ; VSPRITE REG WORK INIT
32D8               495
32D8               496    VSPREGINIT:
32D8 2A 2C 30      497        LD    HL,(VSPREGADDR)
32DB 5D            498        LD    E,L
32DC 54            499        LD    D,H
32DD 13            500        INC   DE
32DE 36 00         501        LD    (HL),00
32E0 01 FF 03      502        LD    BC,1023
32E3 ED B0         503        LDIR
32E5 C9            504        RET
32E6               505
4000               506        ORG   $4000
4000               507
4000               508    ;VSPRITE REG WORK
4000               509
4000               510    VSPREGWORK:
4000               511        DS    1024
4400               512
4400               513    ; VRAM WORK AREA
4400               514    ;  30*29
4400               515
4400               516    VRAM3:
4400               517              DS      1024
4800               518    VRAM2:
4800               519              DS      1024
4C00               520    VRAM1:
4C00               521              DS      1024
5000               522    VRAM0:
5000               523              DS      1024
```

▶ 全機種共通システムインデックス ◀

＊以下のアプリケーションは，基本システムであるS-OS“MACE”またはS-OS“SWORD”がないと動作しませんのでご注意ください。

1986

■86年1月号
- 第13部 リロケータブルのお話
- 第14部 FM音源サウンドエディタ

■86年2月号
- 第15部 S-OS "SWORD"
- 第16部 Prolog-85入門(1)

■86年3月号
- 第17部 magiFORTH発表
- 連載 Prolog-85入門(2)

■86年4月号
- 第18部 思考ゲームJEWEL
- 第19部 LIFE GAME
- 連載 基礎からのmagiFORTH
- 連載 Prolog-85入門(3)

■86年5月号
- 第20部 スクリーンエディタE-MATE
- 連載 実戦演習magiFORTH

■86年6月号
- 第21部 Z80TRACER
- 第22部 magiFORTH TRACER
- 第23部 ディスクダンプ＆エディタ
- 第24部 "SWORD" 2000 QD
- 連載 対話で学ぶmagiFORTH
- 特別付録 PC-8801版S-OS "SWORD"

■86年7月号
- 第25部 FM音源ミュージックシステム
- 付録 FM音源ボードの製作
- 連載 計算力アップのmagiFORTH
- 特別付録 SMC-777版S-OS "SWORD"

■86年8月号
- 第26部 対局五目並べ
- 第27部 MZ-2500版S-OS "SWORD"

■86年9月号
- 第28部 FuzzyBASIC発表
- 連載 明日に向かってmagiFORTH

■86年10月号
- 第29部 ちょっと便利な拡張プログラム
- 第30部 ディスクモニタDREAM
- 第31部 FuzzyBASIC料理法<1>

■86年11月号
- 第32部 パズルゲームHOTTAN
- 第33部 MAZE in MAZE
- 連載 FuzzyBASIC料理法<2>

■86年12月号
- 第34部 CASL & COMET
- 連載 FuzzyBASIC料理法<3>

1987

■87年1月号
- 第35部 マシン語入力ツールMACINTO-C
- 連載 FuzzyBASIC料理法<4>

■87年2月号
- 第36部 アドベンチャーゲームMARMALADE
- 第37部 テキアペ作成ツールCONTEX

■87年3月号
- 第38部 魔法使いはアニメがお好き
- 第39部 アニメーションツールMAGE
- 付録 "SWORD" 再掲載とMAGICの標準化

■87年4月号
- 第40部 INVADER GAME
- 第41部 TANGERINE

■87年5月号
- 第42部 S-OS "SWORD" 変身セット
- 第43部 MZ-700用 "SWORD" をQD対応に

■87年6月号
- インタラプト コンパイラ物語
- 第44部 FuzzyBASICコンパイラ
- 第45部 エディタアセンブラZEDA-3

■87年7月号
- 第46部 STORY MASTER

■87年8月号
- 第47部 パズルゲーム碁石拾い
- 第48部 漢字出力パッケージJACKWRITE
- 特別付録 FM-7/77版S-OS "SWORD"

■87年9月号
- 第49部 リロケータブル逆アセンブラInside-R
- 特別付録 PC-8001/8801版S-OS "SWORD"

■87年10月号
- 第50部 tiny CORE WARS
- 第51部 FuzzyBASICコンパイラの拡張
- 第52部 X1turbo版S-OS "SWORD"

■87年11月号
- 序論 神話のなかのマイクロコンピュータ
- 付録 S-OSの仲間たち
- 第53部 もうひとつのFuzzyBASIC入門
- 第54部 ファイルアロケータ＆ローダ
- インタラプト S-OSこちら集中治療室
- 第55部 BACK GAMMON

■87年12月号
- 第56部 タートルグラフィックパッケージTURTLE
- 第57部 X1turbo版 "SWORD" アフターケア ラインプリントルーチン
- 特別付録 PASOPIA7版S-OS "SWORD"

1988

■88年1月号
- 第58部 FuzzyBASICコンパイラ・奥村版
- 付録 石上版コンパイラ拡張部の修正

■88年2月号
- 第59部 シューティングゲームELFES

■88年3月号
- 第60部 構造型コンパイラ言語SLANG

■88年4月号
- 第61部 デバッギングツールTRADE
- 第62部 シミュレーションウォーゲームWALRUS

■88年5月号
- 第63部 シューティングゲームELFES II

- 第64部 地底最大の作戦

■88年6月号
- 第65部 構造化言語SLANG入門(1)
- 第66部 Lisp-85用NAMPAシミュレーション

■88年7月号
- 第67部 マルチウィンドウドライバMW-1
- 連載 構造化言語SLANG入門(2)

■88年8月号
- 第68部 マルチウィンドウエディタWINER

■88年9月号
- 第69部 超小型エディタTED-750
- 第70部 アフターケアWINERの拡張

■88年10月号
- 第71部 SLANG用ファイル入出力ライブラリ
- 第72部 シューティングゲームMANKAI

■88年11月号
- 第73部 シューティングゲームELFES IV

■88年12月号
- 第74部 ソースジェネレータSOURCERY

1989

■89年1月号
- 第75部 パズルゲームLAST ONE
- 第76部 ブロックゲームFLICK

■89年2月号
- 第77部 高速エディタアセンブラREDA
- 特別付録 X1版S-OS "SWORD"<再掲載>

■89年3月号
- 第78部 Z80用浮動小数点演算パッケージSOR OBAN

■89年4月号
- 第79部 SLANG用実数演算ライブラリ

■89年5月号
- 第80部 ソースジェネレータRING

■89年6月号
- 第81部 超小型コンパイラTTC

■89年7月号
- 第82部 TTC用パズルゲームTICBAN

■89年8月号
- 第83部 CP/M用ファイルコンバータ

■89年9月号
- 第84部 生物進化シミュレーションBUGS

■89年10月号
- 第85部 小型インタプリタ言語TTI

■89年11月号
- 第86部 TTI用パズルゲームPUSH BON!

■89年12月号
- 第87部 SLANG用リダイレクションライブラリDIO.LIB

1990

■90年1月号
- 第88部 SLANG用ゲームWORM KUN
- 特別付録 再掲載SLANGコンパイラ

■90年2月号
- 第89部 超小型コンパイラTTC++

■90年3月号
- 第90部 超多機能アセンブラOHM-Z80

■90年4月号
- 第91部 ファジィコンピュータシミュレーションI-MY

■90年5月号
- 第92部 インタプリタ言語STACK

■90年6月号
- 第93部 リロケータブルフォーマットの取り決め
- 第94部 STACK用ゲームSQUASH!
- 第95部 X68000対応S-OS "SWORD"
- 特別付録 PC-286対応S-OS "SWORD"

■90年7月号
- 第96部 リロケータブルアセンブラWZD

■90年8月号
- 第97部 リンカWLK

■90年9月号
- 第98部 BILLIARDS

■90年10月号
- 第99部 ライブラリアンWLB

■90年11月号
- 第100部 タブコード対応エディタEDC-T

■90年12月号
- 第101部 STACKコンパイラ

1991

■91年1月号
- 第102部 ブロックアクションゲームCOLUMNS

■91年2月号
- 第103部 ダイスゲームKISMET

■91年3月号
- 第104部 アクションゲームMUD BALLIN'

■91年4月号
- 第105部 SLANG用カードゲームDOBON

■91年5月号
- 第106部 実数型コンパイラ言語REAL

■91年6月号
- 第107部 Small-C処理系の移植

■91年7月号
- 第108部 REALソースリスト編

■91年8月号
- 第109部 Small-Cライブラリの移植

■91年9月号
- 第110部 SLANG用NEWファイル出力ライブラリ

■91年10月号
- 第111部 Small-C活用講座（初級編）

■91年11月号
- 第112部 Small-C活用講座（応用編）
- 第113部 MORTAL

■91年12月号
- 第114部 Small-C SLANGコンパチ関数

1992

■92年1月号
- 第115部 LINER

■92年2月号
- 第116部 シミュレーションゲームPOLANYI

■92年3月号
- 第117部 カードゲームKLONDIKE

■92年4月号
- 第118部 オプティマイザO80実践Small-C講座(1)

■92年5月号
- 第119部 COMMAND.OBJ実践Small-C講座(2)

■92年6月号
- 第120部 COMMAND.OBJ2実践Small-C講座(3)

■92年7月号
- 第121部 関数リファレンス実践Small-C講座(4)

■92年8月号
- 第122部 ワイルドカード実践Small-C講座(5)
- 第123部 グラフィックライブラリ GRAPH.LIB

■92年9月号
- 第124部 O-EDIT&MODCNV

■92年10月号
- 第125部 SLENDER HUL実践Small-C講座(6)

■92年11月号
- 第126部 EDIT実践Small-C講座(7)

■92年12月号
- 第127部 MAKE実践Small-C講座(8)

CREATIVE
COMPUTER
MUSIC

Creative Computer Music入門(19)

翻訳の楽しさ

楽器の知識がだいたいついたところで，いよいよお待ちかねの実践編です。自分なりのアレンジをする前に，まずはそれぞれの楽器の特性を活かすことで，原曲のイメージやニュアンスを再現するアレンジを考えてみましょう。これまでの知識を総動員して挑戦してみてください。

Taki Yasushi 瀧　康史

§ CDは惜しまず買え

雑踏のなかでショパンを聴いた。

正確にはショパンを聴いたのではなくって，路頭でダンスを踊っている外人——のように見えた——を見ただけ。

BGMも何もない。まわりの人の気が引きたくて，踊っているのだろうか？　惨めに転がった缶がひとつ。

歩道橋の上からじっと眺めつつ，思いを巡らせてしまう。雑踏の音が微かになりやがて消える。すっと目を閉じまた開ける。そこには，さっきまでとまったく同じ光景が広がっているのに音がない。誰かの話し声，そして車の音……。

3拍半のリズムで踊る人は，不思議とショパンのマズルカを思い出させ，私を興奮させる。ピアノを弾くショパンらしき人物の手のヴィジョンがそこに重なり，路頭のダンサーとシンクロする。ゆったりとしたリズムで心が動いて，うれしくなってしまう。

3拍子の合間の半拍——「ショパンは自分のマズルカを3拍子の合間に半拍入れて弾いた」これはかのシューマンの言葉——が妙にダンスにあう。マズルカはもともとダンス音楽として作曲された……にしても，それを3拍半におく——楽譜上では3拍子——なんて，ショパンもなかなかいいセンスしてる。

3拍子も変拍子もダンス音楽として生まれたって誰かがいっていた。考えながら，手摺に腕をかけて，目を細めてぼっと見てしまうのが好き。もう，待ち合わせなんてどうでもいいや……。

「周りが速すぎるんだよね……」

となりで誰かがいった……いつのまにか，私と同じようにして彼——路頭のダンサー——を見ている男の人がいる。私ときたら，全然気がつかなかった。

⋮

そんなわけで，今月はこういう風に，ショパンのマズルカを紹介してしまいました。

先月ショパンのピアノ協奏曲を久しぶりに聴いてみて，やっぱりショパンの感性っていいな～って思っていた矢先に，お店に行ったらショパン全集全13巻なんて置いてあるのです。思わず買ってきてしまって，今月はそれでなくても財政難なのに，それに追い討ちをかけるかのように，大貧乏になってしまいました。あぁ～このお金，X68030買う資金にしようかと思ったのに。ざ～んねん。

まぁ，そんなわけで，全集っていうのは，たとえ全部を一度に聴かなかったとしても，なかなかよいものです。ほらほら，「こういうのはショパンの○○○によく使われていて……」っていわれたときにすぐ，なるほどって聴けるでしょ。

音楽する人にとって，CDは何よりも大切な教科書ですよね。でも，ただ拝んでちゃ駄目。CDをよく聴くことは，こんな連載を読むよりも，実はずっと身になることなのかもしれません。

「本は惜しまず買え」っていう言葉があるくらいだから，音楽やる人には，CDは惜しまず買えっていうのも，十分当てはまるんじゃないかと思ったわけなのです。でも実は新機種X68030を見て，ちょっと自分のとった行動に自信がなくなってきたのですが……。

それでは，前置きはこれくらいにして，始めましょうか。

楽器の言葉

ピアノにはピアノの言葉があります。

弦楽器には弦楽器の言葉があります。

もちろん，管楽器にも管楽器の言葉があります。

もしもピアノの楽譜をそっくりそのまま管弦楽にもっていってしまったら，はっきりいって，その編曲は興醒めになってしまいます。

原作者の意図を重視するために，そのままもっていく……なんて，そんなのは大嘘です。ピアノの曲を管弦楽で演奏するなら，しかるべき「翻訳」作業を行わなければなりません。なぜって，音の表情は楽器によってまったく違うからです。「減衰」という点のみをみても，ピアノと弦楽器，管楽器では全然違います。まして，音に含まれる倍音列などを考えたらなおさらです。

そういった理由で，今回は音が異なると曲の構成も変わるといったような，いわば「翻訳」のような作業について，主にお話ししていくことにします。

内容が内容なので，この連載のここ何回かで説明したことを一気に復習しなければなりません。1992年12月号の「弦のアレンジ」や1993年3月号でお話しした木管楽器の知識が軒並み必要になるので，そちらのほうを参照しながら内容を把握してくれると幸いです。

た・だ・し。

12月号で弦のアレンジをやったときには，一度にここまで進む予定はなかったので，そこで説明したストリングスについての内容ではすでに足りなくなってしまいました。

別にこの翻訳の作業を「弦」からやらなくてもよいのですが，弦楽器ではそれ単体で曲になっても，管楽器ではどうしても弦楽器の力を借りなければ，まとまりのある長い曲は作りにくい……というような理由から，最初には弦楽器にします。そのためにまず，弦楽器の補講編という感じで，12月号とだぶらない程度にまとめてみましょう。

弦のアレンジ(補講編)

すでに12月号でそれなりに説明してあるので，今回は弦楽「合奏」に重点を置いて話を進めます。

弦楽合奏で特筆すべきことは，弦楽合奏がすべて，ヴァイオリン属で成り立っているということでしょう。

これらのパート別構成は，

ヴァイオリン	2本(1st，2nd)
ヴィオラ	1本
チェロ	1本
コントラバス	1本

と分けることができます。

どれもヴァイオリン属ですから，当然，音色は酷似していて，ほぼ同じ音の高低といっていいほどです。ヴァイオリンはアタックがどちらかというと硬めだとか，それなりの音の違いはありますけれど，それらはここでは大した違いではないとあえていっておきましょう。

もちろん，例外的な構成もありますが，この5本で十分，完全編成といえますので，とりあえずこれだけ覚えておけばよいでしょう。

誤解してはいけませんが，ヴァイオリン1，2とかいってもそれぞれ1本ずつではありません。オーケストラの場合は，同じ楽器で同じ演奏をする人たちが何人かずついます。

人の声には及ばずながら，弦楽器は実に広い音域で，実にさまざまな表情に富んだ楽器といえます。よい演奏では，本当に微かな(***pp***：ピアニッシモ)演奏をしますし(ドヴォルザークの「新世界より」のイントロからAメロディへの推移を聴くとよいでしょう)，盛り上がりではここ一発の***ff***(フォルティッシモ)を実に豊かに演奏することができる楽器です。

また，限度はありますが，実に細かな演奏をすることができる楽器でもあるし，よくいっていたpizzicato(ピッツカート：指で弾く)のほかにも，mute(ミュート：弱音器)，sordino(ソルディーノ：弱音器)，ponticello tremoro(ポンティセロ・トレモロ：普通より駒の近くで弾く)，col legno(コル・レーニョ：弓の背で弦を打つ)など，いま思いついただけでも，これだけ多彩な表現ができるのです。

ヴァイオリン奏者，しかも一級の人はそんなにどこにでもたくさんいるわけではありませんが，一級の奏者がそれぞれの楽器で安定して得られる音域は図1のとおりです。12月号に掲載されたものとは違う部分がありますが，今回は「安全な音域」をテーマに，前回は「発音可能な音域」をテーマにしていますので，そのニュアンスが多少違います。

図1に示した音よりも高音の部分も，曲によってはまれに用いられたりしますが，arco(アルコ：ヴァイオリンなどの弓)で演奏する場合，弦楽器は高音になればなるほど，金切音になってしまい実用的ではありません。これよりも高い音は，エレキギターなどを演奏する人にはわかると思うのですが，ハーモニクス(harmonics：弦振動の分割によって自然に生ずる倍音)を用いることにより可能になります。

ただしハーモニクス音は，やはり使用に関しては例外的で，意図してそれを奏でるとき以外には，使用しないほうがよいかもしれません。

さて，弦楽合奏の和声はやはり四声体において決まります。弦楽器は開離配置だからうんぬんというのは，12月号でさんざん述べたことなので省略しますが，ここでコントラバスについて注意すべき点をいくつか挙げておきましょう。

コントラバス(ダブルベース)は，ヴァイオリン属では最も低い音域をもつ楽器で，その正確な音域はヘ音記号より1オクターブも下に位置するため，記譜では実音より1オクターブ高く表記します。しかし，頭では理解しても，慣れないうちは感覚ではピンとこないものですよね。

コントラバスは一般にチェロと重複させ，そうでないときは，沈黙させるのが通常の使い方です。実はこの「沈黙」というのがポイントで，休符を十分に与えるということは，コントラバスを使ううえでのルールといってもよいでしょう。

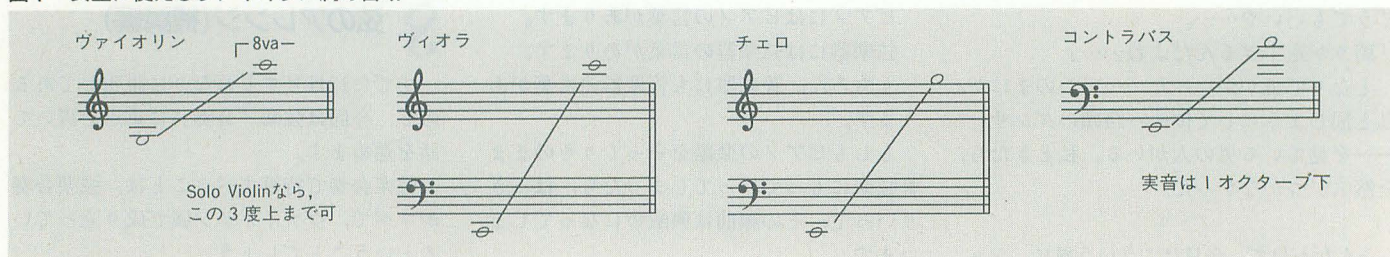
バスブーストなどが盛んにオーディオ機器に用いられる現在ですから，むしろバスを強めるのはブームなのかもしれません。しかし，バスがいつまでも1オクターブ下で重複しているのが，どれほど耳を疲れさせるか考えてみてください。コントラバスの音は，エレキベースの音より1オクターブ下です。たとえロックでも，エレキベースの1オクターブ下に(オクターバか何かのエフェクタを使って)曲の最初から最後まで重複させたらどうなるでしょうか。イメージしてみれば容易にわかりますよね。

楽譜からアンサンブルに入るのは，和声がどのように奏でられているかがいつでもわかるので非常に便利です。そういえば，ロックバンド「X」のYOSHIKIは，楽譜を見ればだいたいどのようにアンサンブルされているかがわかるので，曲としてできあがるまでが早い，といっていましたが(MTRでとって，おかしいと気づいて直して……というバンドは割合多い)，これはこれでたいへんよいことです(実際，楽ですし)。しかし，初心者に限って(ひそかに自分にもいい聞かせているのですが)，コンバスとチェロを同じところに書いて，コンバスが1オクターブ下ということを忘れて使いすぎてしまい，いざ実演させてみると非常に耳が痛かったりすることはままあります。もっとも，DTMですから試行錯誤を繰り返せばどうにかなるのですが……。

そんなわけで，基本は「チェロとコンバスは重複させる」ということであっても，聴き手にそれがあまりにも強い印象とならない程度に，2つの楽器を独立させることが必要になります。

ただし，ここでまた落とし穴があります。

コントラバスは単独で用いられると，非常に乾いた響きになってしまうということです。この解決策として，もしコントラバスのパートが低く，ほかのパートから非常に離れた位置にある場合なら，持続させるか反復させる方法が考えられます。ただしこのとき，これが根音でない場合だと満足な低音の支えができにくくなりますし，これもまたそう簡単には否定できず，十分に低い場合は，穏やかな和声的な経過句の根音を任せることができたりもします。具体的にどの音から下か，とかいうことは調によっても違い，曲調によっても変化します。耳とセンスと経験を活かすべきでしょう(そんな殺生な，といわないでね)。

図1　安全に使えるヴァイオリン属の音域

コントラバスが，1オクターブ高くではなく本来の高さで記譜された場合にその経過句はどのようにみえるか，を常に考えて配置を決めることが，全体を通してのポイントでしょう。

さて，五声部の副音楽経過句(5つの異なったパートによるメロディをもつ曲，と考えてもかまわない)の場合，以下のようなパート分けをするとバランスがうまくとれます。

1．ヴァイオリン1
2．ヴァイオリン2
3．ヴィオラ
4．チェロ1
5．チェロ2とコントラバス1

この分け方は，大管弦などでチェロに余裕がある場合に行う方法で，チェロ2を，音域に余裕があればコントラバスとのユニゾンに，余裕がなければ1オクターブ上で最低のパートを奏でさせ，コントラバスと上のパートをうまく連結させるオーソドックスなテクニックです。

小管弦編成の場合には，チェロを分けてしまったせいで重みがなくなり，高音のチェロ(全体からみると中音域)をつぶしかねないという欠点があるのですが，DTMの場合は，ベェロシティバランスで分けられたチェロをうまくブーストすることにより，バランスをとることができます。

§ ひとつの弦楽器の和音

ヴァイオリン，ヴィオラ，チェロとコントラバス以外のヴァイオリン属の楽器は，ときどき，ひとつの楽器単体で，2ないしは3，4の音で構成された和音を弾くことがあります(これは，12月号で話したつもりだったんですが……ないみたいです。こっちの手違いでした。ごめんね)。

これは技術的に困難なことなので，使用するには特別な注意をしなければなりません。DTMの場合，何でもOKといえばOKなのですが，ここはまあ「らしい」雰囲気を出したい人が読んでください(そんなことは気にしないよ～んって人は，読まなくていいです)。

まず，2音の和声で最も簡単なのはどちらかひとつを開放弦にすることです。つまり，フレットを押さえずそのまま鳴らすことで，当然，音は調弦(図2)だけになります。このことは，同じ弦楽器であるギターを弾ける人には容易に理解できますよね。

では，そうでない人のために。

弦楽器はギターのように，固有振動数をもつ弦が何本か張ってあり，手前から音程が低い順になっています。この固有振動数をもつ弦を調弦といい，この音程はそれぞれの楽器によって異なり，図2のようになっています。これらの張られた弦の固有振動数を決めるファクターは，弦の断面積と長さですから，それぞれの弦は長さは同じで断面積が異なるものだということになります。

このままでは弦の数だけしか音程がないので，なんとか，固有振動数を変えるもうひとつのファクターである「長さ」を変えなければなりません。これが「フレットを押さえる」ということで，ちょうど，どの弦も半音ずつ音が変わるように，押さえるところが決まっているわけです。この，弦を押さえる部分をわずかにスライドさせると，弦楽器特有のポルタメントや，あのビブラートが引き出せるという仕組みになっているわけです。

したがって，和音は，第○弦の第□フレットと，第△弦の第×フレットを押さえて弾く，というようなことになります。当然ながら，フレットを押さえる手の指の長さには限界がありますから，第1フレットと第10フレットなんて届くわけはありませんし，第1弦と第4弦を同時に弾くなんてできっこないのです(同時に聴こえたとしても，うまく回して弾いているのであって，実際には同時に鳴っているわけではない)。

そこで，弦楽器で和音を鳴らすには，できるだけ隣接したフレットで，隣接した弦を求めることになります。これが，弦楽器の和音を鳴らすうえでの困難さなのです。

片方が開放弦なら，フレットは1カ所だけ押さえ，それが隣接する弦なら，ほとんど1音と変わらない手間でできます。そんなところですね。

あれ，ギターはコードバリバリだよ？という方がいるかもしれませんが，ギターは弦が水平に張ってありますが，ヴァイオリン属は弧を描いて張ってあるのです。また，ギターは手もしくはピックで音を鳴らしますが，ヴァイオリンは真っ直ぐに張られた弓で，弧の形をした弦を引いたり押したりして(これがダウンボウ，アップボウとかの指定になる)奏でるのですから，当然，この和音の鳴らし方に違いが出てきます。

さて，片方が開放弦という場合のほかにも，フレット同士が近ければ和音を鳴らすことができます。図3に示すものは，隣接した弦で鳴らすことができ，安心して曲のなかで使えるものです(2音の和音をdouble stoppingといいます)。

仕組みさえわかってしまえば，どれができるかできないかは，導くことができます。図3はヴァイオリンの場合について書いてあるので，ヴィオラ，チェロについては各自で考えてみてください。5度低くするとヴィオラに当てはまり，13度低くするとチェロに当てはまることを覚えておけば，それほど大変なことではないでしょう。

どの図も，その音程より低ければよいという意味ですが，図3-7は，長2度および，短2度のダブルストッピングはこの音以外は避けたほうがよいという意味です。

第2弦の調弦がDなので，ダブルストッピングは高いほうがDより下では，弾くことができません。

想像できるでしょうが，3つの和音はかなりの制約がつきまといます。弦楽器の知識が薄い(いいかえれば，さっきの説明が理解できない)のであれば，最低でも1本の開放弦を含むものだけに限ってしまうのが妥当だと思います。また，開放弦に限ってしまう和音は，力強くて，演奏しやすいので和音にパワーがあります。

フレットは20ほどですから，調弦から数えて半音で20上の音まで鳴ることを覚えてください。当たり前ですが，1本の弦で和音を鳴らせるわけはないので，和音の構成音のどの音がどの弦で鳴るのかを考えながら配置すべきでしょう。図4にとりあえず，

図2　ヴァイオリン属の調弦

音域を書いておきます。

最後に4つの和音ですが，図5にあるとおり，それ自体で開離配置になります。また，前に出てきたように，ヴァイオリン属は弧を描いて弦を張ってあるので，当然同時に音は鳴りませんし，また，和音上の2つのみ同時に持続させることになります。図に示すと，図6のようになり，正確な意味での和音とはいえないかもしれませんが，こういったヴァイオリン属にのみ起きる制約をうまくエミュレートすることにより，DTMでのヴァイオリン属のヴァイオリン属らしさを追求できるとは思いませんか。

§ 弦楽器への翻訳

お待ちかねの，弦楽器への翻訳です。

12月号の内容と，今月号のここまでの内容が全般にわたって出てきますので，そのつもりでいてくださいね。

始める前に，この翻訳の作業ができるようになると，どのようなことができるのか簡単に説明しましょう。

まず，テキストとしてピアノの楽譜を用意します。どうしてピアノ譜かというと，ピアノはたったひとつの楽器で四声体のすべてを押さえることができる数少ない楽器のひとつで，かつポピュラーな楽器だからです。いままで説明したように，弦楽器ひとつで，全体的な形を予想するにはたいへん難しいし，管楽器ではなおさらですよね。いまではDTMがあるので，直接オーケストラの曲を作って打ち込むということもできなくはないですが，オーケストラの曲ともなると，やっぱりいきなりフルサイズの曲を書くのは大変なので，ピアノでその「簡易版」を作り，それを編曲するという方法をとることもできるわけです。もっとも，この作曲法は，作曲者がピアノが弾けなくてはならないという条件がありますけどね。それでも，ピアノのようにただひとつの音色で，すべてを構成するもの(すなわち，比較的ピュアな状態の曲)から，いろいろな音に割り当てていく勉強はやっておいて損はないでしょう。

そうそう。いま思い出しました。

最近は，ピアノを弾く子供たちが好きな曲で勉強できるように，アニメソングピアノピース(楽譜)やら，ゲームミュージックピアノピース(ドラクエとかファイナルファンタジーとか多い)がありますよね。

アニメの曲などは，フルサイズの楽譜はあまり売っていませんから，ピアノから弦楽器へのアレンジを覚えると，ピアノピースから弦楽器へのアレンジができるようになり，便利ですよね(できたら投稿してね……ナウシカとかアリオンの曲って結構好きだな……)。アニメソングの打ち込みをやってみたかったけどフルスコアがなくて苦労していたという人は，気合を入れてください。

さて，当然ですがピアノと弦楽器は音のスペクトルがまったく違います。和声を組むとき，弦楽器は開離配置をするべきですが，ピアノはどちらでもかまいません。また，ピアノはその形状からわかるとおり，隣接してすべての弦が張ってあるため，たった1つの音を弾いてみただけでも，その倍音列の音がそれに共鳴し，わずかに振動します。2つの音を鳴らせば，2音の倍音列の公倍数に当たる音がさらによく振動します。このことが，シンセサイザピアノ(たとえサンプリングピアノでも)が本物らしくない原因なのですが，この事実は，弦楽器へのアレンジにもかなり影響します。

仮に，右手と左手の和音が大幅に離れている曲があったとしましょう(いいサンプルが見つからなかったのですが……)。すなわちこれは，アルトとテナーの差が，1オクターブよりもあるということですから，以前に四声体でお話ししたとおり，こういう和音は内声に穴ができやすくなるのでタブーのはずです。

ところが，ピアノという楽器の場合，この2つの内声(アルト，テナー)のあいだを，弦同士の共鳴によって見事に埋めることができるのです。しかも，この和音は低音域から高音域までの和音の構成音を，見事に踏んでいるわけですから，これはもうダイナミックに響いてくれます。

こういった和音で構成された曲を，まったくそのまま弦楽器にもっていったとしたらどうなるでしょうか。

弦楽器は，共鳴は(若干あるとしても)ありません。だとしたらどうなるかといえば，想像に難くないように，実に貧弱に

図3　ヴァイオリンの2和音の配置

1．この音程までの長3度，短3度

2．この音程までの完全4度，増4度

3．この音程までの完全5度，増5度，減5度

4．この音程までの長6度，短6度

5．この音程までの長7度，短7度

6．この音程までの完全8度

7．長2度，短2度の場合は，必ず1つを開放弦とする。したがって，以下の10種類のみ

ヴィオラは5度，チェロは13度低く移調する

図4　ヴァイオリンの3和音の配置

完全5度，長10度(完全8度＋長3度)の組み合わせである。
この3つの形なら，半音ずつ変化しても可

図5　ヴァイオリンの4和音

図6　実際にはこのようになる

響いてしまうのです。弦楽器には弦楽器の和声配置(開離配置)があるように，この内声の隙間をうまく埋めてやらなくてはなりません。原曲のモチーフを崩したくないならば，ソプラノとバスは変えてはいけないでしょう。

これまでに挙げたことをまとめると，

●内声に大きな隙間のある和音は，ソプラノとバスの位置を変えずに開離配置を行い，あいだを埋める

ということになります。

次に，これと逆のケースを考えてみましょう。ピアノの和声があまりにも密集していた場合はどうなるのでしょうか。まず，密集配置と開離配置の，有利な点と不利な点を挙げてみます。

限られた音の数で和音を組むとき，密集配置に比べて，開離配置は低い音から高い音まで網羅することができます。さっきもいったとおり，和音に広い範囲の音が含まれていればいるほど，厚みが出るのです。これは当然有利な点ですよね。ただし，開離配置の場合，その音が倍音成分をあまり含まない，強いていえばあまり厚みのない音であれば，和音の構成音の隙間が目立ってしまい，逆に美しくありません。これは不利な点です。

いっぽう，密集配置の場合は開離配置に比べて限られた音の数なので，広い音域をカバーはできませんから，当然，厚みがなくなります。これは不利な点です。しかし，金管楽器のような倍音列があまり含まれてない音の場合，和声の構成音に隙間ができないため，構成音同士がうまく絡み合います。こちらのほうは当然有利な点です。

弦楽器がなぜ開離配置にしなくてはいけないのかというと，この2つの配置の特徴から推察されるように，密集配置では音がつながりすぎて，むしろこもってしまうからなのです。

さて，弦楽器の密集配置を，音が高いときと低いときでは，どちらのほうがこもりやすいのでしょうか？　倍音は基音よりも高い周波数になります。ここで，低音域の倍音列は可聴範囲ですが，高音域の倍音列は可聴範囲ではありません。よって，結果的には高音域ではほとんどこもらないのですが，低音域では非常にこもってしまうというわけです。

原曲のピアノ曲で，低音域(それもヘ音記号の下部)で密集した和音があった場合，どうやって避ければいいのでしょうか。これは簡単にいえば，開離配置に置き直すべきですが，もっと確実なのは，さらにひとつ下に「なかに音を含まない」完全8度を作ることで，これによって倍音列のこもりを抑えることができます。具体的にはチェロとコントラバスのオクターブユニゾンを効果的に使うべきでしょう。

これらをまとめると，

●ヘ音記号の下部に密集配置の和音があったときは，バスを変えずに，なかに音を含まない確実な完全8度を1度下に与える

ということになります。

この2つの項目を意識して，例を挙げて考えてみましょう。

まず楽譜1はツェルニーの100番練習曲の17番ですが，右手はさておき，左手はアルペジオによって和声を奏でています。

ここで注意すべき点は，1小節目のバスであるCを強拍では必ず左手がふむことです。このことは(実際このアルペジオにスラーがあればもっとよいのだが)，持続したバスCを表していることになります。そういうことから，楽譜1を弦楽器5本にアレンジした楽譜2では，(ピアノ譜では暗黙にされている)持続されたバスをチェロが押さえています。ここで，コントラバスのピッツカートは，原曲で要求されている強拍に必ず戻るバスCのアクセントを与えています。ヴィオラとヴァイオリン2はオクターブユニゾンで左手のアルペジオを，ちょっと変わった形(2つ目の強拍がCではなくEになって)で奏でています。

ヴィオラの4小節目の1拍目のCはヴィオラの音域を超えているので，コード上の1つ上を奏で，その分，チェロとコントラバスのピッチカートが補充します。

楽譜3，4の例では，駆け上がりの処理をチェロ，ヴィオラ，ヴァイオリン2で引き受けることになっています。これはなぜかというと，仮にこのすべてを音域的に網羅できるチェロだけでは，ピアノでは暗黙のうちに持続される音が切れてしまうため，和声的に不十分になるためです。

そのため各々，和声の構成音で音を持続させてバランスを保っているのです。

2小節目はコントラバスを休ませています。1小節目の和音からの引き潮のようなイメージを出すため，バスを減らしたわけです。ヴァイオリン1はここでは「原曲のモチーフを壊さないアレンジ」であるため，あえて休みにしています。これが，「原曲をさらにゴージャスにするアレンジ」とするなら，何らかのオブリガードや高音部での確実な和音を与えるといいでしょう。

3小節目は，メロディを奏でているヴァイオリン2が音域的にかなり高い部分に移

楽譜1　ツェルニー100番練習曲の17番の冒頭部分(ピアノ譜)

楽譜2　楽譜1を弦楽器5本にアレンジ

動してしまっているため，内声に和声的な欠陥が生じやすくなるため，ヴァイオリン1で補充をしています。

4小節目は，3小節目と同じで，さらに高いところに移行するため，内声の補充だけでなく全体的な和音配置の見直しをしています。

このように，ピアノ譜から弦楽器に置き換える場合，音色の性格の違いから，アレンジにおいてかなりの変化がみられます。

それでも，実はまだ弦楽器はよいほうなのです。どの楽器もヴァイオリン属なので結果的には似たような音色だし，それに発生可能音域ではバランスよく，どこで鳴らしてもまともに鳴ってくれるし。

しかし，これが木管楽器ときたら……。

§ 木管楽器への翻訳

て，その，面倒くさい木管楽器です。

木管楽器(ホルンを含める)だけで長く編曲された曲というのは，非常に珍しいです。曲のなかに，多少の経過句として木管だけの部分があるという程度なら，これはこれでよい印象を与えるでしょうが，これらが長く続くと話は違ってきます。

木管楽器の音の混ざり方は先月さんざんいったとおり，弦楽器に比べて完全ではありません。むしろ，弦楽器……そのヴァイオリン属は音の均一性において完全といってよく，これに比べてしまうと(比べなくとも)木管楽器は，構造上個々の音に分散しがちなのです。

そのため木管楽器でアレンジをする際，気をつけねばならないことは，各楽器を特別にそれぞれの音として扱うことや，楽器の性格に合った小さな経過句をところどころで惜しみなく入れることなどでしょう。

言い換えれば，(このように性格の強い音の)「色彩の混合」をテーマに考えるべきでなく，(性格を活かした音の)「色彩の対照」を重視した経過句を作るべきだといえるでしょう。

こういった理由から，木管楽器(ホルンを含む)の充実した和声の継続は，休みなしに多くの小節に続いてしまうとどうしても重苦しくなってしまいます。それで，木管楽器だけで長く編曲された曲というのは非常に珍しいのです。

例としてここでは，J.S.バッハの「インベンション第1番」(Inventio 1：BWV 772)の最初の経過句を抜粋して，これを木管楽器へ翻訳しています。本来ならばこれに弦楽器を入れるべきですが，ここは木管楽器のアレンジの勉強ですから，あえて木管だけで行います。

とりあえず楽譜5を見てください。

余談として，ちょっと注意をしておきますが，バッハの時代というのはまだ楽譜の書き方が細かいところまで統一されておらず，弾き方が全体的にレガートでいいのかどうかとか，そのようなことについては楽譜にはいっさい書き込まれていません。

トリル(広義なため，モルデントも含む)ひとつとってみても何種類もあり，記号が違っても同じトリルというものもあったりします。

そのため，後世のバッハ研究者たちが，この部分はこう弾くべきだというように主張して楽譜を出版しているため，楽譜によって弾き方が異なることはしばしばです。もっともこの時代は「ピアノ」ではなく，「チェンバロ」ですので……。例のレオンハルトのCDを聴いた方はわかると思いますが，楽器の特質上チェンバロは減衰がピアノよりずっと早く，そのため音をあまりつなげないで弾くのが一般的です。

話をもとに戻し，楽譜1から記号の説明をしてみます。

まず「，」の記号の意味は，指使いの関係上，音が区切れる部分です。

あとは1小節目2小節目にある「〜」(トリル)はMMLでいえば，B8<C8をC32>B32<C32>B32<C8と。ここに細かいMMLで書くとわかりにくいと思ったので省略していますが，本来ならば，最初のC>B<Cはもっと早めに，それでC8につなぐべき

楽譜3 ツェルニー100番練習曲の83番の冒頭部分(ピアノ譜)

楽譜4 楽譜3を弦楽器5本にアレンジ

最後のBはちょっぴり長めにするのが美しいところです。木管楽器へのアレンジはそうは簡単にわからないと思ったので，これにはZMSファイルを載せておきますので参考にしてください。

曲は二2声のインベンションなので，左右1音ずつしか出てきません。基本的構成は右手のメロディを左手に受け渡し，それから右手に受け渡す，というような行為の連続です。

これを木管楽器にアレンジする際に忘れてはならないのは，楽器ひとつひとつの性格を前面に出したアレンジがポイントだということです。

そこで，ここではアレンジの際に，このメロディの受け渡しを，ある楽器からほかの楽器へと受け渡すようにしてみます。

というわけで楽譜6を参照してください。

1小節目の最初のリフは，オーボエが担当します。ここはクラリネットでもいいのですが，バスーンでのメロディの受け渡しをしたかったので，オーボエでまず最初のメロディを奏でます。どうしてもクラリネットで最初のリフを始めたければ，受けをホルンなどでやらせるべきなのですが，ホルンはアクセントがあまりない楽器なので，このようなリフを演奏させると，下手をすると生楽器でもタンギングがかかって音がプチプチ切れてしまうか，逆にモゴモゴするかのどちらかなので，やっぱりやめておいたほうがよいでしょう。

2小節目ではバスーンで受けたメロディをさらにフルートに渡します。ここで，オーボエのメロディは1小節で終わりではなく，楽譜5の2小節目(以後，右手左手ということにする)の最初のDまでは前のメロディに加わると考えてよいので，オーボエにそれをやらせ，左手のG(スタカート)>Gに似たような雰囲気で下のGに落ちます。バスーンは左手のパートを奏でるのですが，下のGの音を長くのばし，これを根音として，C長調のドミナントコードをバスーン，ホルン，クラリネット，オーボエで奏でます。ここのオーボエのハーモニーはあまり美しいとはいえないので，*mp*(メゾピアノ)になりますが，あまり美しいやり方ではありません。全体的に重厚な和声になりますから，それぞれは*p*(ピアノ)ぐらいで演奏するのがちょうどよいというわけです。

また2小節目の根音は，すべてGで補えるので，それに乗った下部のコードはホルン，バスーンでそのまま奏でるとします。ホルンとバスーンのダブテイリングに注意すること。

この小節の最後ではフルートのメロディをクラリネットが受けていることにも，注目すべきでしょう。

3小節目では，すでにメロディの受け流しではなくなっているので，フルートがメロディをすべて吹き，バスーンとホルンで左手のパートを受け持ちます。

クラリネットのリフは，やたらに考えたのではなく，フルートのパートの強拍が経過的になっている点に注目し，その部分だけ抜き出してオクターブ下で奏でるように考慮されています。

オーボエがここで休みなのは，この4つで完成してしまっていることと，オーボエをコードに乗せるとクサるためで，ここでは休みにします。オーボエは4小節目でも休みます。

4小節目では，右手の音の下(くだ)りを最初の小節半分ではフルートが受け持ち，あとの部分をクラリネットが受け持ちます。これは，フルートがこの音域まで下がるとパワーがなくなってしまい(この程度のアンサンブルなどではかまわないのですが)，これに弦楽器などが入ると，はっきりいって消えてしまうため，後ろ半分では，多少デクレッシェンドして右手のメロディの強拍から作られたメロディを1オクターブ上で奏

楽譜5　バッハ「インベンション第1番」(BWV772)より抜粋

楽譜6　楽譜5の木管アレンジの一例

リスト1 「インベンション第1番」より（原曲）

```
 1: (i)
 2: .comment J.S. Bach  Inventio 1 (BWV 772) For GS
 3:
 4: (b1)
 5:
 6: (m1,5000) (a MIDI1,1)
 7: (m2,5000) (a MIDI2,2)
 8:
 9: /  SC-55 Set Up
10:
11: .ROLAND_Exclusive $10,$42={$40,$00,$7F,$00}                / SC-55 Init
12:
13: .SC55_V_Reserve $10={10,10,0,0,0,0,0,0,0, 0,0,0,0,0,0,0}/ Voice Reserve
14:
15: / Max                    7    7      7 127···
16: /                   Macro Char Pre-L Lev Time Del.FB Send Lev.To Chorus
17: .SC55_Reverb $10={   0,   5,     5, 90,  100,    80,                100}
18:                                                           / Reverb set
19:
20:
21:
22: /*       初期設定
23: (o60)
24:
25: (t1) @i$41,$10,$42 @e40 @1 116 @v120 @u75 o4 q7 /* Piano R
26: (t2) @i$41,$10,$42 @e40 @1 116 @v120 @u75 o3 q7 /* Piano L
27:
28: (t1) rcdefdecq4g8q7<c8c48>b48<c48>b16<c8
29: (t2) r2rcdefdec
30:
31: (t1) q4d>q7gab<c>abg<q4d8q7g8g48f#48g48f#16g8
32: (t2) q4g8q7>g8r4rgab<c>abg
33:
34: (t1) q4eq7agfegfagfedcedf
35: (t2) l8q4<c>q7b<cdq4e>q7gab
36:
37: (t1) edc>ba<c>b<dc>bagf#agb
38: (t2) <q4c>q7ef#gab<c4
39:
40: (p)
```

リスト2 「インベンション第1番」より（木管アレンジ）

```
 1: (i)
 2: .comment J.S. Bach  Inventio 1 (BWV 772) For GS Wind Arrange By Kohju
 3:
 4: (b1)
 5:
 6: (m1,5000) (a MIDI1,1)
 7: (m2,5000) (a MIDI2,2)
 8: (m3,5000) (a MIDI3,3)
 9: (m4,5000) (a MIDI4,4)
10: (m5,5000) (a MIDI5,5)
11: (m6,5000) (a MIDI6,6)
12:
13: /  SC-55 Set Up
14:
15: .ROLAND_Exclusive $10,$42={$40,$00,$7F,$00}                / SC-55 Init
16:
17: .SC55_V_Reserve $10={3,3,3,6,6,0,0,0,0, 0,0,0,0,0,0,0}  / Voice Reserve
18:
19: / Max                    7    7      7 127···
20: /                   Macro Char Pre-L Lev Time Del.FB Send Lev.To Chorus
21: .SC55_Reverb $10={   0,   5,     5, 90,  100,    80,                100}
22:                                                             Reverb set
23:
24:
25:
26: /*       初期設定
27: (o60)
28:
29: (t1) @i$41,$10,$42 @e40 @74 116 @v120 @u75 o4 q8 /* Flute    1,2
30: (t2) @i$41,$10,$42 @e40 @69 116 @v120 @u75 o4 q8 /* Oboe     1,2
31: (t3) @i$41,$10,$42 @e40 @72 116 @v120 @u75 o4 q8 /* Clarinet 1,2
32: (t4) @i$41,$10,$42 @e40 @71 116 @v120 @u75 o3 q8 /* Basoon   1,2
33: (t5) @i$41,$10,$42 @e40 @61 116 @v120 @u75 o4 q8 /* Horns    1,2
34: (t6) @i$41,$10,$42 @e40 @72 116 @v120 @u75 o4 q8 /* Clarinet 1,2
35:
36: (t1) r1
37: (t2) rcdefdecq6g8q8<c8c48>b18<c48>b16<c8
38: (t3) r1
39: (t4) r2rcdefdec
40: (t5) r1
41: (t6) r1
42:
43: (t1) @u90rgab<c>abg<q6d8q8g8g48f#48g48f#16g8
44: (t2) q6d8q8@u50>g4.g8<d8>b8<d8
45: (t3) r8@u70'd4.b'@u80r>gab<c>abg
46: (t4) q6g8q8@u40'g2..>g'
47: (t5) o3r8@u40'd2..b'
48: (t6) r1
49:
50: (t1) q6eq8agfq7egfaq8gfedq7cedf
51: (t2) r1
52: (t3) @u50l8o4e4efg4cd
53: (t4) @u60l8q6 c>q8b<cdq6e>q8gab
54: (t5) @u60l8q6<c>q8b<cdq6e>q8gab
55: (t6) r1
56:
57: (t1) edc>ba<c>b<d @u70q7c8a8f#8g8
58: (t2) r1
59: (t3) e4@u70a@u75bl16<@u80c>bagf#agb
60: (t4) <q6c>q8ef#gab<c4
61: (t5) <q6c>q8ef#gab<c4
62: (t6) o4@u50l8q6r4>ab<c>aab
63:
64: (p)
```

でます。

下りてくるメロディを受けるため，クラリネットはクレッシェンドします。経過的に続けていたオブリガードは，1stクラリネットで1オクターブ上げ，メロディとうまくつなげ，2ndクラリネットがその続きを，しかも弱拍をもとにしたメロディで補うことにします。この補いは，バスーン，ホルンがユニゾンで相変わらず左手を奏でてるため，中音域の音がなくなってしまうためです。

以上，原曲の輪郭をくずすことのないように，私なりにアレンジをしたものです。これはほんの一例なので，ほかにもいろいろできるでしょう。その際，今日やった問題を忘れないように。なぜ3～4小節ではオーボエがないかは，もはや記すべくもないでしょう……が，念のため説明すると，オーボエが入るとしたらクラリネットのユニゾンになってしまい，またクラリネットとオーボエのユニゾンはタブーであるからです。試しに実際にやってみればわかるでしょう。

もちろん，ここでのアレンジは「翻訳」という作業をもとにしているので，「自分なりのアレンジ」としては除外して考えています。

原曲とアレンジしたものの両方のZMSファイルを掲載しておきますから，参考にしてください。

§ 最後に

これでおわかりになったでしょうが，このように性格の異なる楽器間で行う編曲は，主にその楽器の特色をよく知ることから始まります。

その点，弦楽器は楽なのですが，木管楽器はどれも個性が強くて強くてなかなか苦しませてくれます(でも，その分面白いけどね)。

次に考えるのは，これらを混ぜてやるにはどうするのかとか，さらに金管楽器も混ぜる場合とかについてでしょう。

それらについては，来月やってもいいんですが一気に覚えようとすると頭に入らないので(とはいえ，金管楽器のほうが木管楽器よりは楽)，まだこれはお楽しみということでとっておきましょう。

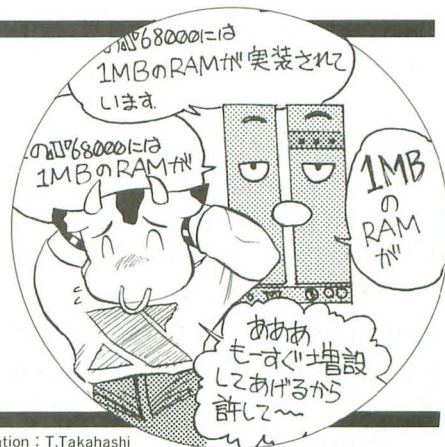
今月の知識を駆使して，何かピアノ譜から小管弦でアレンジしてもいいかな～とも思ったのですが，いつか「オマケ」という形でつけましょうか？

では，今月はこれでおしまい。

(で)のショートプロぱーてぃ――その43

あやしいパソコン大作戦

Komura Satoshi 古村 聡

今月のショートプログラムは，ちょっと実用，ちょっとお遊びの２本です。ようやく訪れた春のせいでしょうか？ (で)氏の言動もちょっとゆるみがち。でも，押さえるべきポイントはしっかり押さえているから安心ですね。

illustration：T.Takahashi

ねーねー，皆さん見ましたか「ゆみみみっくす」。すごいですよね。

「ゆみみみっくす」は残念ながらX68000のゲームじゃなくてMEGA-CD(メガドライブのCD-ROM)のゲームなんですけど，竹本泉(そう，あの「あおいちゃんパニック」描いたマンガ家さんですね)さんの絵が，絵が，絵が〜。全画面でぐりぐり出てくるんですよ，もうすごいんです。ほとんどゲームというよりは，マルチエンディングのアニメビデオなんです。私はデモを見て，あまりのすごさにメガCD持ってないのにソフトだけ買ってしまったんですよ。いやいや，すごいすごい。

でね。私ね，考えたんです。次の世代のパソコンの条件ってこういうのがやっぱりほしいなって。

「○ーラームー○のオープニングが画質も大きさも落とさないで再生できること！」

この原稿が本になるころには，もう第１部が終わっちゃってますけど。

だったらビデオ買えばいいじゃんといわれるかもしれないけど，やっぱりアニメ番組くらいの画質でゲームがしたいじゃないですか。きっとキャラクターへの思い入れも違ってくると思うんですよね。

まだまだ，現在あるどのパソコンもちょっとここまでは遠いけど，いつかこんなパソコンが出るといいですよね。X68000シリーズは結構射程距離内にある気もするんだけど，RAM容量の問題が大きいかな。誰か，X68030で挑戦してくれると面白いんだけどな〜。

なにはともあれ，がんばれパソコン，がんばれX68000シリーズなのであります。まる。

動かないよと思う前に⑹

今月掲載したHC.BASってちょっとクセのあるプログラムなんですよね。っていうのもコースエディタってユーザーの自由度が高いぶん，使うときに気をつけなきゃいけないことがあるんですよね。で，今月は先手を打って，質問電話にかかってきそうな事柄を書いておきます。

●コースからはみ出しても終わらないよ

そう，自分の車がコースからはみ出してもゲームが終わらなくなっちゃうことがあるんですよね。でもこれ，原因は自分のミスなんですよ。よーく，画面を見てください。はみ出したところの色が写真と違ってないですか？

そうこのゲーム，コースの中，つまり車の走れるところをpaint関数で塗りつぶしているので，コースの始めと終わりの間が抜けちゃうとpaintがはみ出してしまって，コースを外れても走れてしまうようになるんですね。今度は気を付けて，隙間ができないようにちゃんとコースを描き直してくださいね。

●コースエディットができなくなった

で，コースエディタでコースを描き換えたいと思った。ところが！ コースを作ったあとでセーブしてしまってコースエディタが使えない……。

うーん，同じ名前でセーブすると前のファイルは消えちゃいますからねー，本当は常日頃からいろんな名前でセーブするように心がけたほうがいいんですよ。ディスクをケチっちゃだめですってば。

で，このプログラムの場合は100行の部分だけ元と同じように打ち直せば大丈夫です。さぁ，今度は違う名前でセーブしましょうね。

●敵車をなくしたい

これも前と同じ。ってことはこのゲームの場合最低３つの名前でセーブしたほうがいいわけですね。で，この場合は300行を書き換えるだけでOKです。

●gotoの飛び先がないといわれる

renumしたでしょ。renumって行番号を切りよく並び替えてくれるから便利だけど，goto文を使っているときには飛び先を変えてくれないから，注意しなくちゃいけないんですよ。

で，今回のプログラムの場合，コースエディタやコースレコードとかが行番号に依存しているので，できるだけ行番号は変えないで。変える場合は自分の責任でやりましょうね(もちろん，そのときにも別の名前であらかじめセーブしておくといいのは，いうまでもないですね)。

なお，今月のぱーてぃハンズはお休みさせていただきます。ごめんなさい。

HOW MANY?

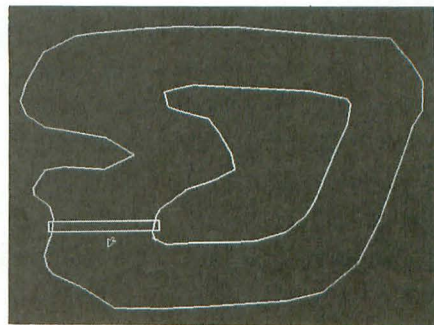
それでは今月の１本目のプログラムにいきましょう。北海道の薄井さんのプログラムで，実装RAM容量を報告してくれるHOWRAM.Sです（リスト１）。どうぞ。

HOWRAM.S for X68000

（要アセンブラ，リンカ）

北海道　薄井広樹

このHOWRAM.SはX68000に実装されているRAMが何メガバイトあるかを画面に表示し，同時に終了コードとして返すというプログラムです。

このプログラムのリストはアセンブラのソースという形で書かれています。プログラムリストをエディタで入力してから，アセンブラ，リンカを使って実行ファイルを作ってください。たとえば，エディタに標

HC.BASコースコンストラクション

準添付のED.X，アセンブラ，リンカにそれぞれAS.X,LK.Xを使うのであれば，

A>ED HOWRAM.S

でリストを入れてから，

A>AS HOWRAM.S

A>LK HOWRAM.O

として，実行ファイルHOWRAM.Xを作ります。また，CV.Xにかければ〝R〟ファイル，HOWRAM.Rにすることももちろんできますから，お好みでどうぞ。で，使い方は，

A>HOWRAM

でRAM搭載量が表示されます。たとえば全部で4MバイトのRAMが載っているマシンなら，「このX68000は，4MBのRAMを実装しています」という具合に表示されます。

また，終了コードでメガバイト数，この場合なら「4」という値が返ってきます。

また，バッチファイルで使っていて，こっそり終了コードだけがほしい，なんてこともありますよね。そういうときには，/Nオプションをつけて，

HOWRAM /N(または−N)

とすると画面には表示させずに(画面表示を抑制する，なんていいますよね)終了コードだけを返してくれます。

このプログラムではRAMの搭載量は，SRAMの$ED0008番地にあるRAM最終番地＋1のデータを流用しています。そのマシンのRAM最終番地＋1が仮に＄◇◇◇◇◇◇ならば，0番地から始まって＄(◇◇◇◇◇◇−1)番地まで，＄◇◇◇◇◇◇バイト分のメモリが載っているわけです。この値を$100000(1024×1024)で割り，メガバイト数を出しています。さらにこの値を文字に変換したものをメッセージとともに画面（正しくは標準出力）に出力しています。最後にこの値を終了コードに返してEXIT2で終わりというわけです。

ん〜。なかなかシンプルなプログラムですね。実際にこのプログラムを使う場面としては，たとえばAUTOEXEC.BATなどでメモリ実装量に応じて実行するアプリケーションを変えるなどというのが考えられますね。ほら，アニメバリバリのゲームを作って，RAM容量が足りなかったらコマを落とすとかね。

最後に，このプログラムはフリーウェアとするそうです。どんどんみんなで使いましょうね。そうそう，ちゃ〜んと作者の方に敬意を払って，作者の名前もいっしょに入れてあげてくださいね。

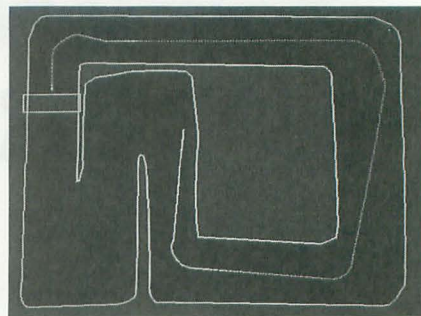
走れ走れラジコンカー

それでは続いて2本目のプログラムの紹介。今月の2本目は東京都の中村さんのプログラムで，ちょっとお得なカーレースゲームHC.BASです（リスト2）。どぞっ。

HC.BAS for X68000

（要X-BASIC）

東京都　中村俊之

画面上をドットが動いていくライトサイクルゲーム感覚のCAR GRANDPRIXゲームです。このリストの短さでコースのコンストラクションキットつき(コースを作るためのプログラムね。英語からそのまんまだけど)という，なんともお得なプログラムなんですよ。このプログラムを実行するにはジョイスティック(とマウス。X68000ならマウスは標準でついてくるから大丈夫だろうけど）が必要です。では，使い方。

HC.BAS

まずはHC.BASのリストを打ち込みます。そして，間違いがないことを確認したらいったんセーブしてください。

RUNするとプログラムはコースエディタモードに入ります。ここではまず，コースを作ります。このコースエディタモードはマウス専用で，画面の左上に番号とマウスカーソルが表示されます。

リスト1　HOWRAM.S

```
 1:
 2: *<HOWRAM. S>
 3: *          :as howram / lk howram (好みで cv howram )
 4:
 5:          .include        doscall.mac
 6:          .include        iocscall.mac
 7:
 8: *上の2つのインクルードファイルを持っていない方は、
 9: *下のマクロ/定数定義行の「*」を外してください。
10:
11: *DOS      macro    num
12: *         .dc.w    num
13: *         endm
14:
15: *IOCS     macro    num
16: *         moveq.l #num,d0
17: *         trap     $0F
18: *         endm
19:
20: *_PRINT            equu     $FF09
21: *_FPUTS            equ      $FF1E
22: *_EXIT2            equ      $FF4C
23:
24: *_B_LPEEK          equ      $84
25:
26:          .text
27:          .even
28:
29: start:
30:          lea.l    stackbss(pc),sp
31:          moveq.l #$01,d7                              *画面表示フラグ=1
32:          tst.b    (a2)                                *引数存在チェック
33:          beq      main
34: spcskip:
35:          addq.l   #1,a2
36:          cmpi.b   #' ',(a2)           *スペース
37:          beq      spcskip
38:          cmpi.b   #$09,(a2)           *タブ
39:          beq      spcskip
40:
41:          cmpi.b   #'-',(a2)           *スイッチか?
42:          beq      option
43:          cmpi.b   #'/',(a2)
44:          bne      option_err
45: option:
46:          addq.l   #1,a2
47:          cmpi.b   #'?',(a2)
48:          beq      usage
49:          cmpi.b   #'N',(a2)
50:          beq      disp_flg_clr                        *画面表示しない
51:                                                       *オプション
52:          cmpi.b   #'n',(a2)
53:          bne      option_err
54: disp_flg_clr:
55:          clr.l    d7                                  *画面表示フラグ=0
56: main:
57:          lea.l    $00ED0008,a1                        *($ED0008)=
58:                                                       *RAM最終番地+1
```

最初にコースの外周を作っていきます。マウスカーソルを動かしつつ，左ボタンをカチカチと押していくと，押した場所と場所が直線で結ばれていきますので，それで描いていってください。外周が切れているとゲームするときにうまくいかない(ペイントしたときにはみ出してしまうため)のでちゃんと始点と終点がつながるように気をつけてくださいね。

次は内周。外周を描き終わったら右ボタンを押します。そうすると画面左上の数字が2になりますから，同じようにマウスで描いていってください。

そして，ゴールの四角を描きます。これはゴールの四角の対角線の2点をクリックしてください。画面左上の数字は4，5が表示されます。

最後に画面に6の数字が出たらスタート地点をクリックします。

これでコースのエディットが完了しました。画面上にリストが出てきますから，上から順にリターンキーを押してください。

では，お待たせしました。これでやっとゲームができますよ。

r.上でリターンするとコンピュータが，さっきエディット画面で作ったコースを描きます。その間，画面の真ん中あたりに表示される数字は，そのコースのこれまでのあなたが出した最高タイムです。

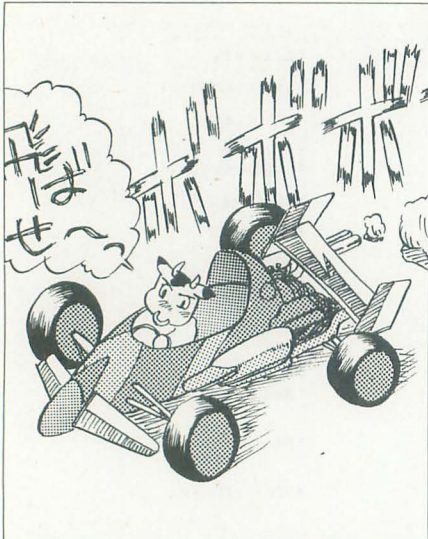
コースの上で白い1ドットのHCカーを表す，ちょっとずつ上に進んでるドットがわかりますか？　それがあなたの車です。ハンドルはジョイスティックのレバーの右左で，ブレーキがAボタン，Bボタンがアクセルになっています。これを使って少しでも早くゴールに着きましょう。壁に激突すると初めからやり直しです。

ゴールに着くと，またリストがずらーっと出てきます。例によってリターンキーを押していってください。するとさっき自分が走ったのと同じ走りの敵車が出てきます。そこからは，赤い軌道の過去最高速車が自分と競争します。自分との戦い，ということになるわけですね。ゴールに向かってGO！ GO！ しましょう。

そうそう，自分で取っておきたいな，という記録が出たときにはセーブしておきましょうね。そのときにはファイルネームを変えるのを忘れないように。でないと二度

```
 59:          IOCS     _B_LPEEK                    *ここがキモだ。
 60:          moveq.l  #20,d1
 61:          lsr.l    d1,d0                       *÷$100000(=1MB)
 62:
 63:          move.w   d0,d2                       *メガバイト数退避
 64:
 65:          lea.l    MB(pc),a0                   *バッファ下端
 66:                                               *(用意の仕方がセコい)
 67:          clr.l    d1
 68: itoa_loop:                                    *数値→10進文字
 69:          divu.w   #10,d0                      *÷10
 70:          swap.w   d0                          *D0.W=余り
 71:          move.w   d0,d1
 72:          add.b    #'0',d1                     *文字に直す
 73:          move.b   d1,-(a0)                    *バッファへ
 74:                                               *(桁を1つ増やしつつ)
 75:          swap.w   d0
 76:          andi.l   #$0000FFFF,d0               *商が0でなければ
 77:          bne      itoa_loop                   *次の桁へ
 78:
 79:          cmp.b    #0,d7                       *フラグが寝てたら
 80:          beq      no_disp                     *表示しない
 81:
 82:          pea.l    rammsg(pc)
 83:          DOS      _PRINT                      *表示
 84:          addq.l   #4,sp
 85: no_disp:
 86:          move.w   d2,-(sp)                    *メガバイト数を返す
 87:          DOS      _EXIT2
 88:
 89: usage:
 90:          move.w   #2,-(sp)                    *stderr
 91:          pea.l    usagemsg(pc)                *ヘルプ表示
 92:          DOS      _FPUTS
 93:          addq.l   #6,sp
 94:          move.w   #$FFFF,-(sp)
 95:          DOS      _EXIT2
 96:
 97: option_err:
 98:          move.w   #2,-(sp)
 99:          pea.l    opterrmsg(pc)               *オプションが違います！
100:          DOS      _FPUTS
101:          addq.l   #4,sp
102:          pea.l    usage2msg(pc)               *正しいオプション
103:          DOS      _FPUTS
104:          addq.l   #6,sp
105:          move.w   #$FFFF,-(sp)
106:          DOS      _EXIT2
107:
108: rammsg:
109:          .dc.b    'このX68000は、
110: MB:
111:          .dc.b    'MBのRAMを実装しています。',$0D,$0A,$00
112: usagemsg:
113:          .dc.b    'RAM実装量検知ツール for'
114:          .dc.b    ' X68000 version 1.02 By '
115:          .dc.b    '薄井 広樹(神楽岡電脳公司)'
116:          .dc.b    $0D,$0A,$0A
117:          .dc.b    'RAMの実装量を表示し、'
118:          .dc.b    '終了コードとしても返します',$0D,$0A
119: usage2msg:
120:          .dc.b    $0D,$0A
121:          .dc.b    '書式:',$09,$09,'HOWRAM [option]',$0D,$0A
122:          .dc.b    'オプション:',$09,'/N (/n・-N・-nでも可)'
123:          .dc.b    $09,':画面に表示しない',$0D,$0A
124:          .dc.b    $09,$09,'/? (-?でも可)',$09,$09
125:          .dc.b    ':ヘルプを表示',$0D,$0A,$00
126: opterrmsg:
127:          .dc.b    'オプションの指定が違います。'
128:          .dc.b    'もう一度確認してください。',$0D,$0A,$00
129:
130:          .even                                *無いとハマる。
131:
132: stacktop:
133:          .ds.b    20          *このぐらいあればいいかな、なんて。
134:
135: stackbss:
136:
137:          .end
```

リスト2　HC.BAS

```
  10 /*  HC CAR GRANDPRIX.
  20 /*                                1992/10/31  おくの撲滅運動会
  30 float x, y, a=0.23#, va
  40 int cc=1,ec=992,mc=65535,v,s,t,k,gx,gy,c,f,x1,y1
 100 mouse(1):mouse(4):msarea(5,5,505,505): goto 1000
 300 dim oz(5000),hz(5000)={ 1500 }:goto 2000
 999 /*------------------ editor
1000 dim dx( 200 ) ,dy( 200 )
1010 int ox, oy, p=0, i, j, L, r, d
1020 screen 1,3,1,1
1030 console , , 0
1040 repeat
1050     mspos(i,j)
1060     msstat(d,d,L,r)
1070     locate 0,0
1080     print p;
1090     if L <>0 then {
1100         pset( i,j,mc )
1110         dx(c)=i
1120         dy(c)=j
1130         c=c+1
1140         p=p+1
```

とコースエディットできなくなっちゃいますよ(ま，もとのリストと違うところを打ち込み直せばいいんだけど。ショートプロのいいところですよね)。

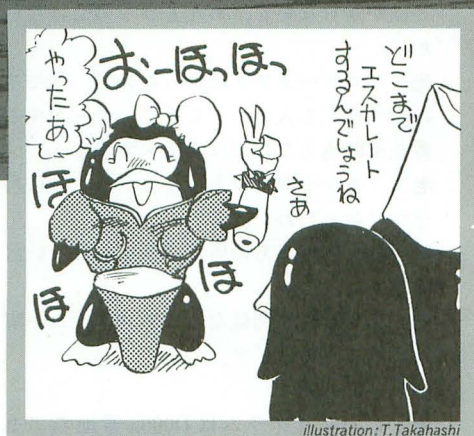
うーん，これが，ジョイスティックを右に倒すと右回転。左に倒せば左回転。ゴール出てすぐのときはいいんですけどね。車の進む向きによっては感覚と逆になっちゃうんですよ。これってやっぱりラジコンカー感覚ってやつなんでしょうね。むずかしいよ〜。楽しく遊ぶにはコースの幅は広くしておいたほうがいいですね(最初むずかしいコースばっかり作ってゲームにならなかったヤツ)。あ，それとコースはスタート地点から，まず縦に進むように作るのがコツみたいです。

どうしてもうまくゲームができなかったらリストを変えてしまうのも手のひとつかもしれないですね。参考までにステアリングの変数がa，加速がvvです。

う〜ん，短いわりにはお得なプログラムなんだけど，ちょっとアラが目立つかなぁ。配列が足りないため，点が多すぎるとプログラムが止まったりとか，軌跡が364までしか記録されないというのは，しかたないかもしれない。でも，出てすぐゴールに戻れちゃうのはなんとかならなかったかな。次はそのへんも考えてくれるとうれしいね。

さ〜て，人にお願いばっかりするのもなんだし，そろそろ私もなんか作ろうかな。アニメーションがしたい。その前にX68030がほしい。煩悩は人を行動的にさせますね(どういう行動じゃそりゃ)。ううーん……また来月っ。

```
 1150           if p = 6 then box( ox, oy, i, j, mc ):L=0
 1160           if p = 2 or p = 4 then line( ox, oy, i, j, mc ):p=
p-1
 1170           ox=i
 1180           oy=j
 1190           repeat
 1200               msstat( d,d, L,r )
 1210           until r+L=0
 1220       }
 1230       if r<>0 then {
 1240           if p=1 or p=3 then {
 1250               dx( c )=0
 1260               c=c+1
 1270               p=p+1
 1280           }
 1290       }
 1300 until p=7
 1310 screen 2,0,1,1
 1320 mouse( 2 )
 1330 print
 1340 print " 100 dim d(420)={ 0"
 1350 for t = 0 to c
 1360     print
 1370     print 110+t;
 1380     for b = 0 to 6
 1390         print using ",###";dx( t );
 1400         print using ",###";dy( t );
 1410         t=t+1
 1420     next
 1430     t=t-1
 1440 next
 1450 print " }"
 1460 print "r."
 1470 locate 0,0
 1480 end
 1999 /*---------------- corse write.
 2000 for i=0 to 5000:next
 2010 screen 1,3,1,1
 2020 console ,,0
 2030 hc=1: oc=1
 2040 h=hz(0): va=0: v=270:k=1:c=3:f=0:t=1
 2050 x2=d(c-2):y2=d(c-1)
 2060 repeat
 2070     x1=x2: y1=y2
 2080     x2=d(c): y2=d(c+1)
 2090     c=c+2
 2100     locate 29,16
 2110     if x2=0 then {
 2120         f=f+1
 2130         x2=d(c): y2=d(c+1)
 2140         c=c+2
 2150         print h
 2160         if f=2 then {
 2170             box( x2, y2, d( c ), d( c+1 ), mc )
 2180             x=d(c+2): y=d(c+3)
 2190             paint( x, y, cc )
 2200             cls
 2210         }
 2220     } else line(  x1 , y1 , x2 , y2 , mc  )
 2260 until f=2
 2270 gx=d(c): gy=d(c+1)
 2280 c=3
 2290 palet(0,ec)
 2300 palet(2,0)
 2310  /*------------ Dynamite spaicy main program.
 2320 repeat
 2380     s=stick( 1 )
 2390     if s=4 then v=v-5 else if s=6 then v=v+5
 2400     s=strig( 1 )
 2410     if s = 0 then {
 2420         va=va-0.01#
 2430         k=0
 2440     }
 2450     if k = 0 and s= 1 then va=va+a else if s=2 then va=va-
a
 2460     oz( c )=x: oz( c+1 )=y
 2470     t=t+1
 2480     if va < 0 then va=0.2#
 2490     pset( hz(c)+1, hz(c+1), 1985 )
 2500     c=c+2
 2510     pset( x, y, mc )
 2520     x=x+cos(v*0.017444#)*va*0.8# : y=y+sin(v*0.0174444#)*v
a
 2530 until point( x, y )=0
 2540 print "time=";t
 2550 if x2>x or x>gx  or  y2>y or y>gy  or  t>h then goto 2000
 2560 screen 2,0,1,1
 2570 console 0,32,0
 2580 print
 2590 print " 300 dim oz(5000),hz(5000)={";t;
 2600 for t=1 to c
 2610     print
 2620     print 310+t;
 2630     for b=0 to 21
 2640         e=oz(t)
 2650         if t<c then print using ",###";e;
 2660         if t>=635 then t=c: b=21
 2670         t=t+1
 2680     next
 2690     t=t-1
 2700 next
 2710 print " }:goto 2000"
 2720 print "r.";
 2730 locate 0,0
 5000 /*+--------------------------------------
 5010 /*|x,y    HCcar X,Y   |a  加速力   |cc corce color
 5020 /*|gx,gy  goal X,Y    |va 現在速   |ec rough color
 5030 /*|s      stick,strig |k  スリップ |mc HCcar color
 5040 /*|v      HCcar 方角  |t  ラップ   |hz() 速い軌跡
 5050 /*|c      t*2 軌跡用  |h  ベスト   |oz() 君の軌跡
```

illustration: T.Takahashi

マシン語カクテル
in Z80's Bar
第41回

今月は，成りゆき上，とぼけた光君がマウスドライバを制作します。ニコニコぷんの全国巡業を終えて復活した，メアリーが登場する今月のZ80's Barは，いつもよりちょっぴりボケボケ。みんな，いくら春とはいえしっかりしてね。

いってきマウス

Shiba Mamoru **司馬 護**

カランコロ～ン♪
源光（以下光）：こんちは～。
ようこ（以下Yo）：いらっしゃ～い。
光：最近暖かくなりましたね。
メアリー（以下メ）：Oh!　ニッポンジンとイギリス人は天気のお話ダイスキですね。
光：メアリー，久しぶりじゃないか。
メ：チョッチュネー。私ニッポンにいたけど。帰ってきたあるよ。
光：なんだか日本語のおかしさに磨きがかかってるみたいだけど。
マスター（以下M）：メアリーはね，ピッコロのアルバイトをしてたんだよ。
光：ピッコロってドラゴンボールの？
Yo：ちがうわよ。ニコニコぷんの営業だって。
光：じゃじゃまるとか出てるやつ？
メ：そうでござール。
Yo：ピッコロってペンギンだったかしらね？
メ：そうでございマース。
長老（以下老）：大陸の人間はいつもおおらかじゃのう。
メ：長老さん，ペンギンは南極大陸にすむとですよ。
老：ところで最近ハヤリの着せかえコーナーはどうなったのじゃ。
光：そういえば，ようこさんチャイナドレス着てる。
Yo：今日はね，町内会でミス春麗大会があるのよ。
M：優勝候補ナンバー1だっていわれて，その気になっちゃったみたいだよ。
Yo：私昔から百烈キックだけは自信あったのよね。
老：よく光君のことをけっとばしとったもんじゃ。
Yo：投げもまんざらじゃないし。
老：彼氏になったら苦労するじゃろうな。
Yo：ってことで，私はミス春麗大会に行ってきまーす。
カランコロ～ン♪

ネズミが出マウス

光：やれやれ，今日はひとりで店番ですね，マスター。
メ：なにいうとる。私がいるでねーか。
光：その万国方言博覧会みたいな日本語，どうにかならないの？
メ：日本全国巡業の旅に行ってたけんね。あきまへんわ。
老：あっ，なんか動いてる。
メ：ぎゃっ～，ネズミー。
光：ドラえもんじゃあるまいし。あらら，気絶しちゃったよ。
M：メアリーも女の子だね。
老：感心しとらんで，ネズミを処理するなり，メアリーを起こすなりしなさい。
M：よしきた。光君，ネズミの処理はまかせた！
光：ネズミねえ。どっちかっていうと，女の子を熱い口づけで起こすほうが得意なんだけどなあ。しようがないか。
老：今日はやけに聞き分けがいいのう。
光：さあて，プログラミング，プログラミング，と。カチャカチャ。
M：？
老：ついに光もボケおったんかいのう。
光：えっ？　マウスを処理するプログラムを作れっていったのはマスターですよ。カチャカチャ。
M：まあいいっか。プログラム作ってくれるんなら文句はありません。
老：ほれ，メアリー，起きんしゃい。
メ：うーん。
M：光君がマウスのプログラムを作ってくれるって。
メ：Oh!　Mouse!
老：しもた。また気絶しおったわい。
M：でもニコニコぷんの営業だったらネズミが苦手っていうのも変だけどなあ。
老：実は相手がネズミだって気づいていなかったのかもしれんぞ。
M：メアリーならありえますね。
光：でーきたっと。
M：メアリーちゃん，起きなさい。
メ：はーい，よい子のみんな元気かな？ピッコロはと～っても元気よ。
老：だめじゃ，こりゃ。

説明しマウス

M：マウスってなにげなく使っているけど，どうやって入力してるんだろう。
光：お約束の質問ありがとう。
M：どういたしまして。
光：えっと，大きく分けると2通りのマウスがあります。
老：野ネズミとドブネズミじゃな。
光：違います。バスマウスとシリアルマウスの2種類です。
老：ねこバスなら知っておるが，バスマウスとはこれいかに？
光：PC-9801なんかでは一般的に使われているマウスですね。
M：パラレルマウスって言葉も聞いたことあるけど。
光：バスマウスと同じって考えて差し支えありませんよ。
M：それじゃあシリアルマウスは？
光：Xシリーズ全般に使われているマウスですね。
メ：パラレルとシリアルっていうくらいだから並列と直列なんでしょ？
M：おおっ，メアリーが標準語をしゃべった！
メ：ステージでは標準語をしゃべりますから。
老：きっとネズミを見たショックでおかし

くなったんじゃろう。

光：あのう，マウスの話は。

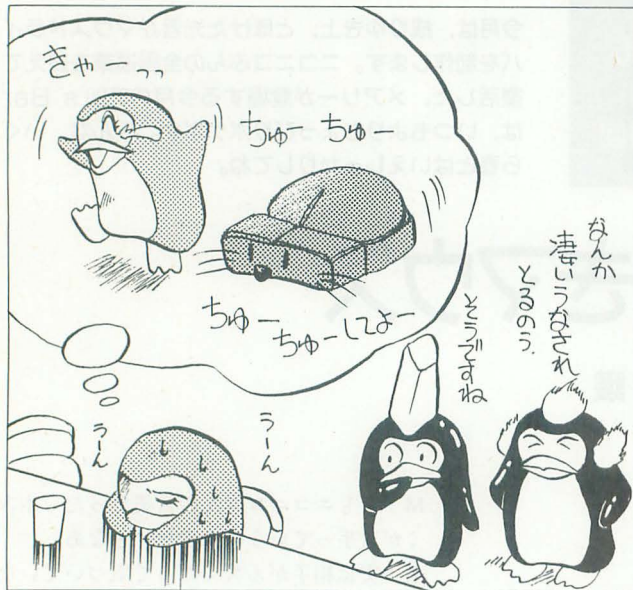
M：続けてください。

光：で，今回はシリアルマウスを取り上げるということで，こっちは詳しく説明しましょう。

メ：さっきの答えがまだよ。

光：そうそう，シリアルマウスはSIOを使って，データを1ビットずつ送るんだ。メアリーちゃん正解。

メ：えっへん。

光：で，シリアル通信だから，ボーレートが決まっているし，データの順序も決まっています。

老：ほう。

光：ボーレートは4800bps，データは3バイトで，その内訳は表1のようになっているんですよ。

M：これって全機種共通なんですか？

光：うーん。Xシリーズは同じだと思うんですけど，ほかのシリアルマウスはどうでしょう。

M：要するに，今月のプログラムはX1用なんですね。

光：そういうことになります。

解説しマウス

M：で，どんなプログラムなんですか？

光：えっと，マウスドライバってことになります。

老：ああ，プラスドライバと。

メ：それはマイナスドライバ。よい子のみんなは真似しちゃだめよ。

M：どんなふうに使えばいいんですか？

光：えっと，マウスを使うプログラムに組み込んで，好きなように使ってもらえばいいんですけど。

老：組み込むといってもアセンブラを使うとはかぎらないし，めんどうなのではないか。

光：リロケータブルに作ったので，メモリの隙間にでも入れてもらって，コールする番地を間違えなければ大丈夫ですよ。

老：リロケータブルとはにくいところに目をつけおったわい。

M：あんまりうちの店では馴染みのない言葉ですね。

光：アドレスに関係なく，プログラムが実行可能ってことです。

M：というと？

光：このプログラムはD000H番地でアセンブルされてますけど，アセンブルし直さなくても任意の番地で実行できるんですよ。

M：だったら全部のプログラムをリロケータブルにすれば便利じゃないですか。

老：絶対番地を指定するとリロケータブルではなくなるのじゃ。

光：そう。JP命令やCALL命令が入ると特定の番地でしか動作しなくなってしまうんですよ。

老：Z80のプログラムでJPやCALLなしで作るとなるとかなり大変じゃろう。

M：なるほど。

光：まっ，自慢じゃないですけど今回のプログラムは，リロケータブルにするためにSIOのセットルーチンなどかなり汚いですけどね。

老：どれどれ，うっ，データをすべてスタックに放り込んでおるわい。

光：はは，16バイトぐらい軽い軽い。

M：どんなデータを送ってるんですか？

光：SIOにはマウスの設定に合わせて，データは8ビット，スタートビット1ビット，ストップビット1ビット，クロックは16分周するなどですね。

メ：テキストは「Z80ファミリーハンドブック」になっております。

光：断っておくけど，CQ出版だからね。NHK出版協会じゃないからね。

M：なんでクロックを16分周するんですか？

光：4800bpsにするためですよ。

老：CTCも設定しておるようじゃが。

光：チャンネルBをカウンタモードのタイムコンスタント26で使ってます。

M：ということは？

光：えっと，CTC側でチャンネルBは2MHzでクロックをカウントするでしょ。タイムコンスタントは26でしょ。SIO側で16分周してるから，

$$1/(2\times10^6)(\mathrm{Hz})\times26\times16$$
$$=208\times10^{-6}(\mathrm{SEC})$$

表1

レジスタ	意味	説明
A	ステータス	D_0 スイッチ1 (1→on, 0→off) D_1 スイッチ2 D_2 未使用 D_3 未使用 D_4 Xのオーバーフロービット (X>127) D_5 Xのアンダーフロービット (X<−128) D_6 Yのオーバーフロービット (Y>127) D_7 Yのアンダーフロービット (Y<−128)
H	Y方向の移動量	−128〜127
L	X方向の移動量	−128〜127

リスト1

```
D000 C5 D5 F3 01 A2 1F 3E 47 : D4
D008 ED 79 3E 1A ED 79 21 C1 : 06
D010 03 E5 21 00 07 E5 21 00 : 16
D018 06 E5 21 00 05 E5 21 44 : 5B
D020 04 E5 21 00 02 E5 21 00 : 12
D028 01 E5 21 18 18 E5 01 93 : B0
D030 1F 16 08 E1 7C ED 79 7D : 7D
D038 ED 79 15 20 F6 FB 3E 05 : CF
D040 ED 79 AF ED 79 3D 20 FD : D5
D048 3E 05 ED 79 3E 02 ED 79 : 4F
D050 26 00 16 03 AF ED 79 ED : 41
D058 78 1F 30 F8 0B ED 78 03 : 32
D060 6F E5 15 20 EF E1 7D E1 : B7
D068 67 D1 7B D1 C1 C9       : 0E
---------------------------------
SUM: 6B C4 44 86 48 D7 F5 A8 DD75
```

$$1/208 \times 10^{-6} \fallingdotseq 4800(\text{bps})$$

って計算になるわけです。

M：なるほど。

光：あとはライトレジスタ(WR) 5を立ち下げると，データがSIOから受け取れる仕組みになっている。と，こんなんでいかがでしょう。

老：プログラムが短いわりに説明がずいぶん長かったのう。

光：ファミリーLSIを2つ使うデバイスドライバですからね。

おおボケしマウス

老：これを前回の爆弾掃除人に利用することはできるかね。

光：不可能ではないだろうといっておきましょう。

老：ってことはかなり難しいってことかのう。

光：ええ，試しにやってみたんですけどね。あのプログラムはS-OSのFLGETっていうルーチンを使って，カーソルを表示していたんですよ。

老：ほう。カーソル点滅1文字入力というやつか。

光：ええ，それを横取りしようとすると，カーソル点滅は消せないし，そのほかのキーの関係もあって，文字入力も必要だし，ちょっと難しいかなって。

老：試しに作ってみたんじゃろ。

光：一応，やってみました。爆弾掃除人自体を斜め入力対応にしておけばよかったって後悔しましたよ。

老：なるほど斜め入力か。マウスでは当たり前じゃからのう。

光：爆弾掃除人をソースで入力した人だったら，比較的簡単にマウス対応にできると思いますよ。

老：斜め入力対応にして，カーソルをマウス入力にするだけだからのう。

光：やりがいのあるチャレンジですね。

カランコローン♪

M：おや，ようこちゃんが帰ってきた。

光：ミス春麗にはなれました？

Yo：あのう……。

老：準ミスくらいにはなれたんじゃろう。

Yo：いえ。

M：審査員特別賞だったとか。

Yo：先週だったのよ。ミス春麗大会。

光：さすが，ミス(Mistake)Z80's Bar。

M：平凡なオチでしたね。

メ：よい子のみんなは真似しないでね。

—つづく—

リスト2

```
0000                1 ;     MOUCE.SYS    ( Relocatable )
0000                2 ;
0000                3 ;         OUT     A=STATUS,H=X,L=Y
0000                4 ;         Break   A,HL only
0000                5 ;
0000                6 ;     by Hikaru Minamoto
0000                7
D000                8           ORG     $D000
D000                9
D000               10 SIO       EQU     $1F90
D000               11 CTC       EQU     $1FA0
D000               12
D000               13 MOUCE
D000 C5            14           PUSH    BC
D001 D5            15           PUSH    DE
D002 F3            16           DI
D003               17           ;
D003               18 SETCTC
D003 01 A2 1F      19           LD      BC,CTC+2
D006 3E 47         20           LD      A,$47    ; Ch=B
D008 ED 79         21           OUT     (C),A
D00A 3E 1A         22           LD      A,$1A    ; Time constant
D00C ED 79         23           OUT     (C),A
D00E               24 SETSIO
D00E 21 C1 03      25           LD      HL,$03C1; Data=8bit,Enable Receive
D011 E5            26           PUSH    HL
D012 21 00 07      27           LD      HL,$0700
D015 E5            28           PUSH    HL
D016 21 00 06      29           LD      HL,$0600
D019 E5            30           PUSH    HL
D01A 21 00 05      31           LD      HL,$0500
D01D E5            32           PUSH    HL
D01E 21 44 04      33           LD      HL,$0444; Clock*16,STOP=1bit
D021 E5            34           PUSH    HL
D022 21 00 02      35           LD      HL,$0200; INT_Vector=00(turbo)
D025 E5            36           PUSH    HL
D026 21 00 01      37           LD      HL,$0100
D029 E5            38           PUSH    HL
D02A 21 18 18      39           LD      HL,$1818; RESET * 2 (1ﾂﾊ Dummy)
D02D E5            40           PUSH    HL
D02E               41           ;
D02E 01 93 1F      42           LD      BC,SIO+3
D031 16 08         43           LD      D,8
D033               44 SIOLOOP
D033 E1            45           POP     HL
D034 7C            46           LD      A,H
D035 ED 79         47           OUT     (C),A
D037 7D            48           LD      A,L
D038 ED 79         49           OUT     (C),A
D03A 15            50           DEC     D
D03B 20 F6         51           JR      NZ,SIOLOOP
D03D               52           ;
D03D FB            53           EI
D03E               54           ;
D03E               55 MAIN
D03E 3E 05         56           LD      A,5      ; WR5    BC=SIO+3
D040 ED 79         57           OUT     (C),A
D042 AF            58           XOR     A        ; > Hi
D043 ED 79         59           OUT     (C),A
D045               60 WAIT
D045 3D            61           DEC     A
D046 20 FD         62           JR      NZ,WAIT
D048               63           ;
D048 3E 05         64           LD      A,5
D04A ED 79         65           OUT     (C),A    ; WR5
D04C 3E 02         66           LD      A,2
D04E ED 79         67           OUT     (C),A    ; > Low
D050               68           ;
D050 26 00         69           LD      H,0
D052 16 03         70           LD      D,3
D054               71 GET
D054 AF            72           XOR     A        ; RR0
D055 ED 79         73           OUT     (C),A
D057 ED 78         74           IN      A,(C)
D059 1F            75           RRA
D05A 30 F8         76           JR      NC,GET   ; DATA EMPTY
D05C 0B            77           DEC     BC       ; SIO+2
D05D ED 78         78           IN      A,(C)
D05F 03            79           INC     BC       ; SIO+3
D060 6F            80           LD      L,A
D061 E5            81           PUSH    HL
D062 15            82           DEC     D
D063 20 EF         83           JR      NZ,GET
D065               84           ;
D065 E1            85           POP     HL       ; Y = H
D066 7D            86           LD      A,L
D067 E1            87           POP     HL       ; X = L
D068 67            88           LD      H,A
D069 D1            89           POP     DE       ; STATUS = A
D06A 7B            90           LD      A,E
D06B               91           ;
D06B D1            92           POP     DE
D06C C1            93           POP     BC
D06D C9            94           RET
OBJECT CODE END D06D
```

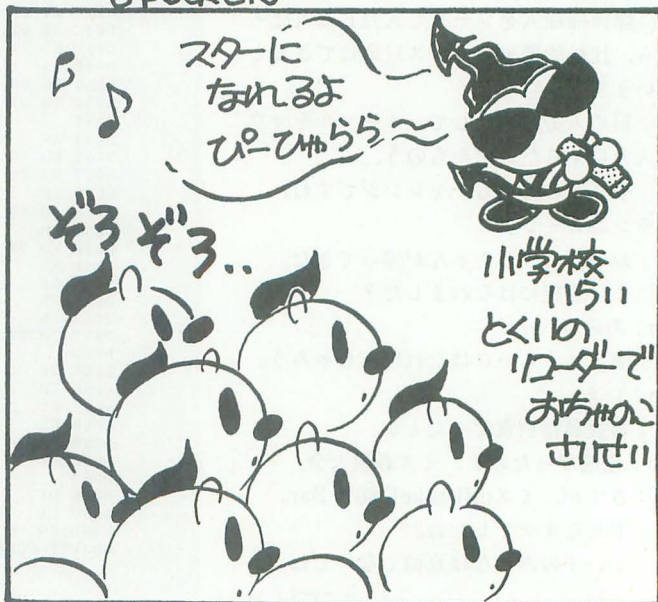
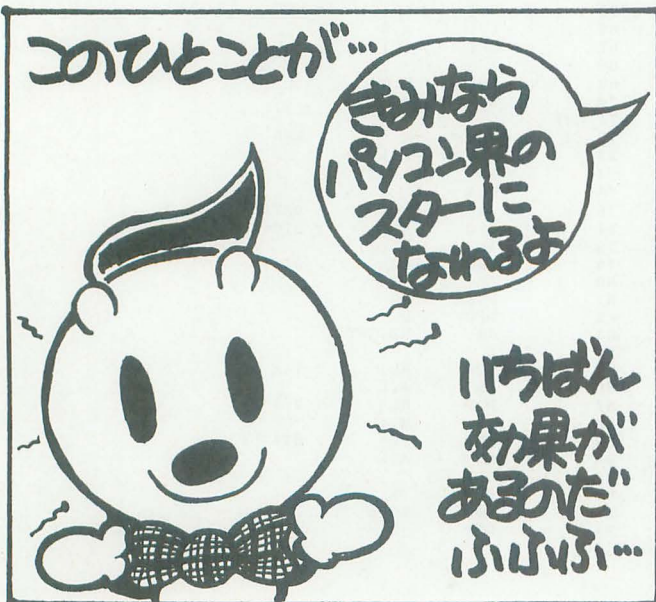
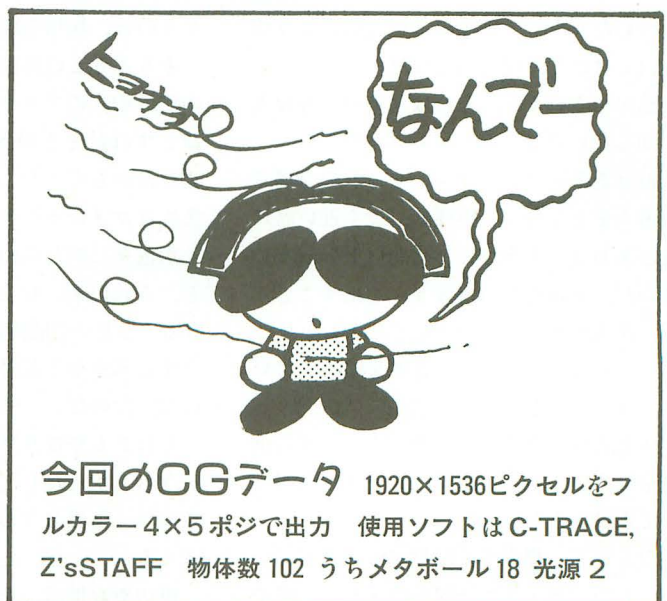
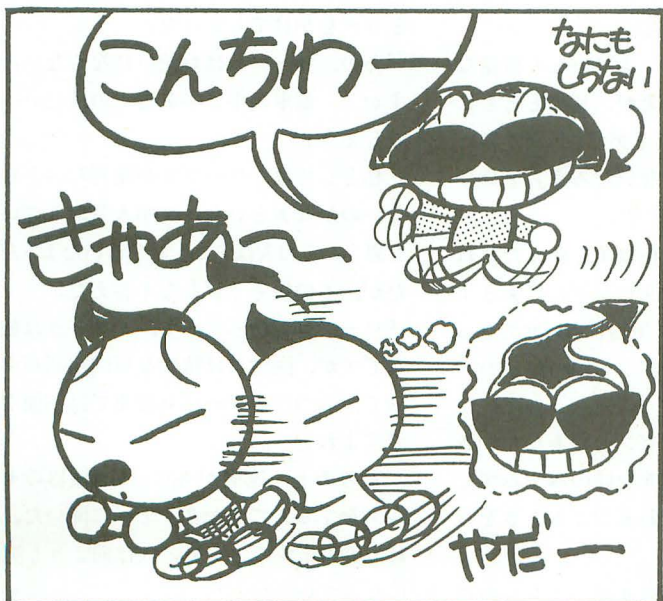
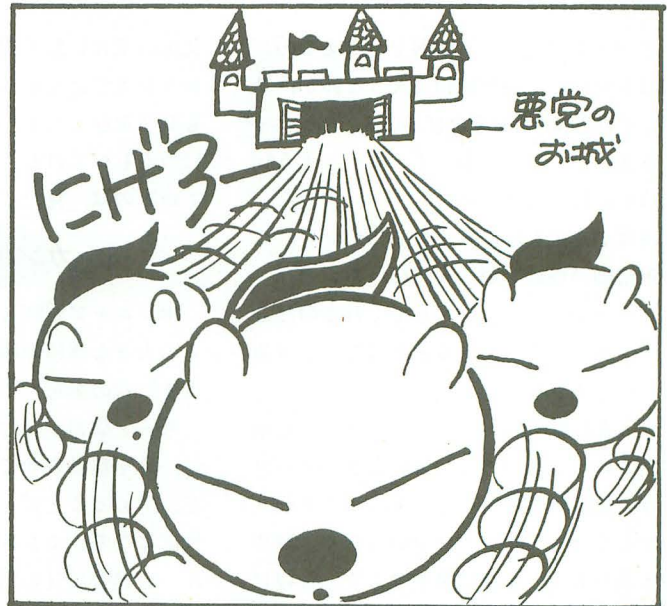
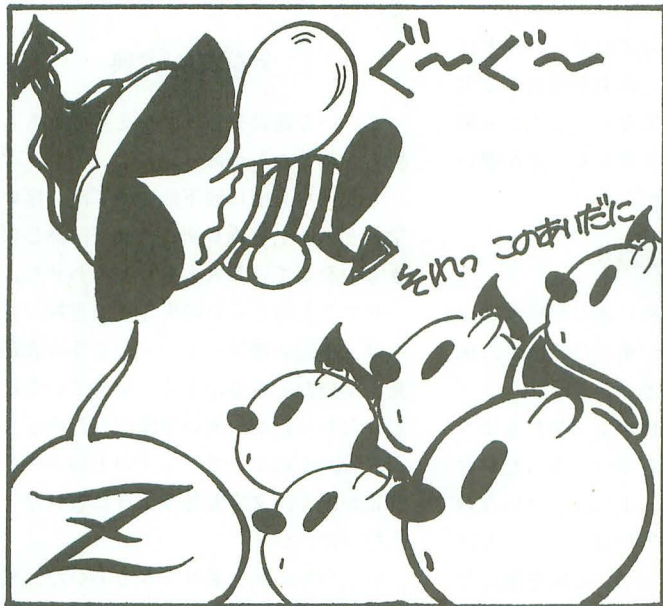
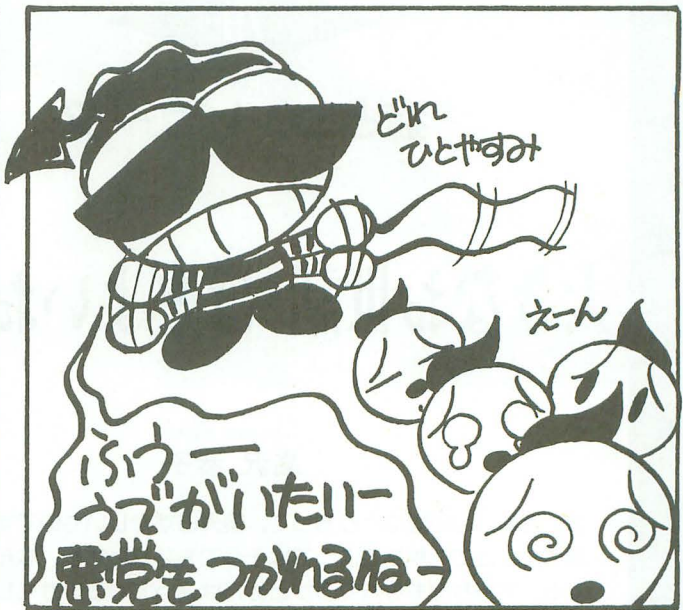
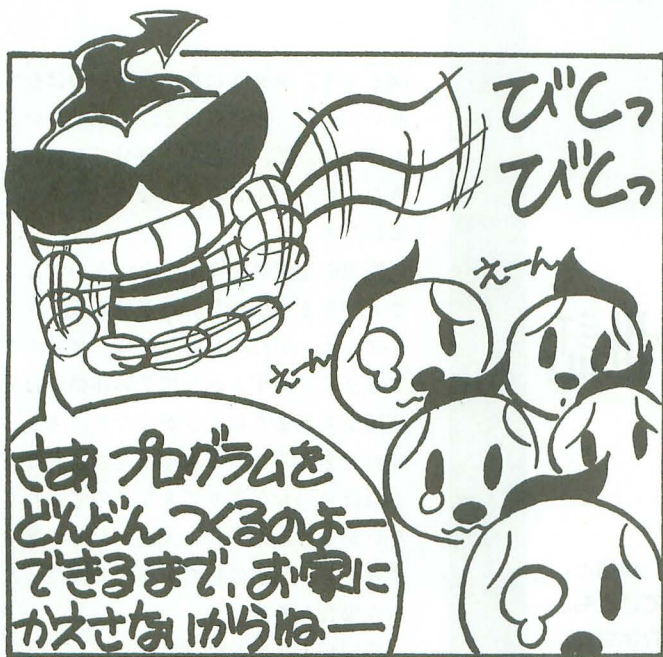
ANO THER CG WORLD

ふふふ…
あたしは悪党

プログラミングの好きな子どもを あつめて ひともうけを たくらんでいるのよー

いまどきの子どもは とても めぐまれているので…
フン!
おかし
いちごポッキー
おこづかい
100
ストII
ゲームソフト
こんなものには見むきもしない…
6 pockets

このひとことが…
きみならパソコン界のスターになれるよ
いちばん効果があるのだ ふふふ…

スターになれるよ ぴーひゃらら〜
ぞろぞろ…
小学校いらいとくいのリコーダーでおちゃのこさいさい

今回のCGデータ 1920×1536ピクセルをフルカラー4×5ポジで出力　使用ソフトはC-TRACE, Z'sSTAFF　物体数102 うちメタボール18 光源2

猫とコンピュータ

大きなお世話・ほどよいお世話

Takazawa Kyoko
高沢 恭子

おつきあいの距離のとり方って，案外むずかしいものです。親切と思ったことでも相手によってはそうでもなかったり，なんてことも。でも，気持ちのもち方でうけとめ方は違ってくるのかもしれません。

スパティフィラム。名前をおぼえるのに何日もかかった観葉植物の大きな鉢が，リビングの窓ぎわで日をあびている。ホンニャアは迷わずそこまで歩いていき，きょう何回目かの昼寝をはじめた。

猫はなんでもだまってするのがいい。この無言劇の静けさが，人間にいろいろなイマジネーションをもたらしてくれるのだと思う。何かをうったえるとき以外，猫はあまり声を出さない。

犬は吠えるときのほかにも，ひとりでしゃべっていることがある。暑いときハァハァいうのはしかたないとしても，歩きまわりながらモグモグ，ブツブツいったり，期待や不満があるとすぐに鼻をならす。昼寝に入るにも，ドサッと倒れこみながらタメ息をついたりする。グチの多い人間のようで面白いけれど，現実的なのだ。

私がそういうと，それは名犬や上等な犬を知らないのだとトオルがいった。

彼はあるクラシックコンサートの会場で，盲導犬をともなった男性と，ごく近い席に並びあわせたそうだ。2時間以上の演奏のあいだ，犬はただの一度も声を出すことなく，主人である男性の足元でときおりシッポをふりながら，ともに音楽を鑑賞しているようだったという。

「休憩時間になるとネ，男の人のヒザに前足をかけてじゃれるんだけど，それでもけっして声は出さないんだよ」

なるほど，想像するだけでも立派で，いじらしい。ディズニー漫画のプルート君や，父母の家にいた世間知らずのシェパードばかりが犬ではなかった。麻薬を発見する犬もソリをひく犬も，忠実なかしこさで人間につくしてくれる。そう考えると猫が働いているのは，見たことがない。

カンがたより

ホンニャアが眠るために選んだ場所には，その大きな鉢植の植物が葉をひろげて，床にまだらの影をつくっていた。

もっと猫のからだに日がよく当たるように，鉢を移動させてやろうかと思ったけれど，やめることにした。また，よけいなお世話になる。ホンニャアが自分で，いちばんいい具合に日の当たっている床を選んだのに，わざわざコンディションを変えてしまうのは，無用な親切だ。

そもそもこの鉢植も，よけいなお世話なのだ。狛江のアニキ夫妻が，長男とともにおとずれたときのオミヤゲで，「おマエ，花なんかもらったことないだろう」という失礼なコメントつきだった。

水芭蕉に似た純白の花は美しかったけれど，それ以来，せまいリビングには大きすぎるずうたいの植物に，気温にあわせてじょうずに水をやる苦労をずっとつづけるハメになったのだ。

それでもやはりプレゼントのあたたかさがうれしかったし，生きた緑のみずみずしさには，迷惑をうわまわるここちよさをおぼえる。

親切やお世話，思いやりが，こちらの意図どおり，全面的に功を奏するのはむずかしいことだ。迷惑だけど，ありがたい。ありがたいけど，迷惑だ。自分でもまわりの人たちに，たくさんそんな思いをさせているのだろう。

電車の中で席をゆずるという小さなことさえ，うまくいかないことがある。タイミングとカンがたいせつなのに，グズグズしてしまう。ゆずるべき相手の体格が自分の座席よりあまりに大きかったり，ちょっと離れたところにいる人だったり。あるいはそれほどのお年寄りでもないので失礼ではないかと考えたり。一大決心をして席をゆずったら，つぎの駅で下車する人だったという場合もある。

親切が大きなお世話にならないようにするのは，けっこうたいへんだ。

デジタル古典

よけいな進言をして，かえってこちらが教えられたことがあった。

もう何年も前に地下鉄の中で，分厚い医学書とノートを両ヒザにのせて，熱心に書きものをしている人ととなりあわせた。

40すこし前くらいの男性で，書物の背中には「小児病理学」といったような表題が見えていた。電車はとくにゆれているほどではなかったが，寒い季節だったのでその人も着ぶくれていて，ヒザの上のノートは安定が悪く，文字も思うように書けないふうだった。

いくつかの駅を通りすぎるあいだ，私はそのようすを横目で見ていたのだが，とうとう声をかけてしまった。

「あの，ワープロはお使いにならないんですか？　電車の中で字を書くのはたいへんでしょう」

私としては，ワープロが便利なのを知らないのかなぁという，文明人気取りでいたと思う。その人はすこしおどろいたようだったが，不愉快そうでもなく答えた。

「ワープロも持っていて，ときどきは使うんですが，医学には特殊な専門用語が多いので，ふつうのワープロ辞書では不便なんですよね」

そのあと，遠慮がちに「OA機器のセールスをする人ですか？」と私に聞いた。あわてて否定すると，その人は前よりも笑顔になっていった。

「それと，ドイツ語や外来語がひんぱんに使われますからね。アルファベットだけではダメです。文字の大きさも問題で，全角の文字配列では間のびしてしまうし，半角では小さすぎます。どっちにしても文章の中でのバランスがとても悪いです」

生半可な知ったかぶりで人に何かをすすめたことと，それがセールスとまちがわれたことが，ともに恥ずかしかった。新しい文明の利器といっても，生まれて間もない商品なのだから，すぐに誰の役にでもたつとはかぎらないのだった。

あれからずいぶんたって，それぞれに改良もすすんでいるが，あの人はもう満足できる状態でワープロかパソコンを使っているだろうか。それとも，こんどは別の理由で手書きのしごとをつづけているだろうか。

いくら医学の専門用語に使われる文字でも，第2水準の漢字にはおさめられているだろうから，一度ひろって登録しておけばすぐに使えるはずだ。独，仏の文字が出るワープロ辞書だってあるだろう。

それでもやっぱり，研究にはノートとペンがいちばん好きという人にとっては，機械は異物かもしれない。

先日の新聞記事に，日本の古典文学をデータベース化する計画が，いよいよ完成の段階に入ったとあった。もう6年も前から文部省国文学研究資料館が進めてきたもので，1995年には完成の予定だそうだ。

岩波書店の「古典文学大系」全百巻におさめられている，日本の古典560作品をデジタル化して，そのなかに登場するあらゆる字句を自在に検索できるようにするものだという。

たとえば「あはれ」という言葉を検索すると，作品ごとに，その件数と，どのページの何行目に用いられているかが，その個所の本文とともに示されるのだ。

『古事記』『万葉集』『源氏物語』から歌舞伎の台本まで，3千万もの文字が，要求にあわせて瞬時に引き出される。ひとつの言葉や文字だけでなく，「花」と「月」などというように，あわせて用いられた場合の検索もできる。

古典文学の研究にたずさわる人たちにとって，作品の中の1つひとつの語句の用例をあつめることは，もっとも基礎的で重要なしごとのようだ。これはたいへんな手間がかかり，この作業そのものが一生の課題になってしまうこともあるという。

そうした時間をはぶいて，コンピュータから統計的な情報を得たうえで，独自の課題やテーマを追究することができるなら，研究の方法として能率的である。もうすでに内部ではテスト運用をしていて，3年後には大学の研究室などのパソコンを資料館のホストコンピュータにつないで，オンライン検索ができるようにするのだそうだ。

ラーメンオフの感激

大量の古典文学を，コンピュータによって，すばやく文字単位で検索できるようにしてもらえるなんて，いそがしい研究者のかたたちはどんなに助かることか。

でもいままで，このしごとを手作業でコツコツとつづけ，それをテーマとしてきた人はどんな心境だろう。2年後にはデータベース化が完了して，どんな言葉も一瞬でリストになる。自分のテーマが別のところで完成したのを見たら，つぎのテーマをさがすことを最大のテーマにしなくてはならなくなるのだろうか。

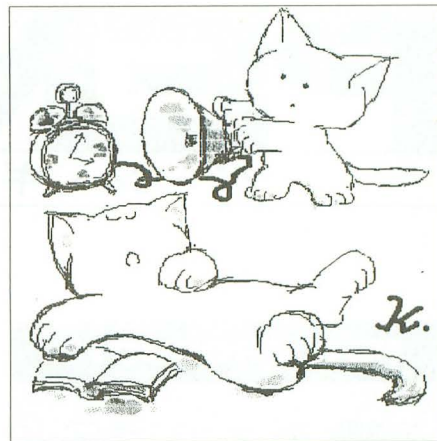
古典のデータベース化はよけいなお世話とは思わない。ただ，文学や物語は，ひとりの人の感性で一貫して読みすすめていくことでそれぞれのとらえ方が生まれてくる。部分としての言葉が先にあるのではなく，全容によって部分の言葉の位置づけができる。データベースで検索されたものと，人間がとらえるものは同じではない。

ほどよいお世話はむずかしい。からだを競技用の矢で射抜かれたオナガガモのさわぎもそうだったけれど，結果的にはみんながホッとしたのでよかった。

先日は，夫が普通乗用車の免許を取ったことを，若い人たちがお祝いしてくれるといううれしいお世話もあった。

夫は三重県でしごとをするようになってから，県内の通信ネットにもいくつか加入させてもらっているそうだ。その中のひとつで，Mori-NETという小さなネットのメンバーが，お祝いの会を開いてくれたのだ。題して「ラーメンオフ」。

どこの誰かわからないオジサンのためにみんなが喜んでくれる。これがネットワーカーのフシギなつながりだ。

シスオペは中古自動車をあつかう会社に

illustration：Kyoko Takazawa

つとめるかたで，ハンドルネームをガードナーさんという。会社の中にホスト局をおいて，顧客との情報連絡に役立てているそうだ。

このオフ会に参加してくれたのは，当日急にこられなくなった人もいて，けっきょく女性2人をふくめた6人。みんな20代の元気いっぱいの人たち。このあつまりがとても新鮮だったのは，展開の予測がつかないところにもあったらしい。

集合は夜の8時。場所は県内の名張市という，夫のマンションから電車で30分くらいのところ。もちろん夫にとって初対面のかたばかり。8時から9時まで1時間ほど，四日市などからの参加予定者をのんびりと待つ。おナカがすいてくる。それからみんなで向かったのが，「天理ラーメン」という名物ラーメンの店がある，奈良県天理市だった。到着したのが夜の10時で，ラーメンオフ会はここの屋台での開催となった。すこしびっくりしたが，あまりに空腹だったこともあって，ともかく味は最高だったそうだ。

さて，これで閉会なのかなと思ったら，ふたたび名張市にもどって，こんどはカラオケ屋さんに入った。すでに11時20分，それから楽しむこと約1時間。夫もおつきあいに1曲うたったそうだ。クルマで上野市のマンションまで送ってもらい，ドアをあけたのが深夜の1時をまわっていた。

あまり予定にこだわらない，深夜ももものとはしない若さ，元気さ。ひとり年長者の夫だったが，ともに楽しんですごせたことに感謝したという。

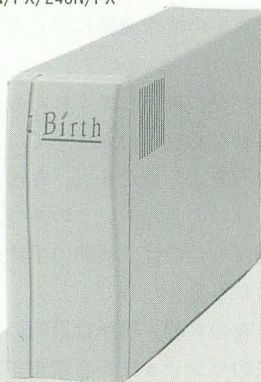
ちょっと刺激的な，思いがけないうれしいお世話だったのだ。

PENGUIN INFORMATION CORNER

ペ・ン・ギ・ン・情・報・コ・ー・ナ・ー

NEW PRODUCTS

X68000用SCSIハードディスクドライブ
Birth-170N/FX/240N/FX
日本アルトス

Birth-170N/FX/240N/FX

日本アルトスは，X68000用SCSI外付けハードディスクを発売した。

今回発売されたものは「Birth-170N/FX」「Birth-240N/FX」の2機種で，主な仕様は以下のとおり。

●Birth-170N/FX
　記憶容量：168Mバイト
　平均アクセスタイム：16.5ms
　キャッシュメモリ：232Kビット

●Birth-240N/FX
　記憶容量：240Mバイト
　平均アクセスタイム：16ms
　キャッシュメモリ：256Kビット

両機ともSCSIインタフェイスを使用，重量は約1.6kg，外形寸法は50mm(幅)×280mm(奥行)×130mm(高さ)となっている。

なお，接続ケーブル，終端抵抗（ターミネータ）は同梱されている。

価格は「Birth-170N/FX」が99,800円，「Birth-240N/FX」が138,000円（ともに税別）である。

〈問い合わせ先〉
日本アルトス(株)　☎03(5820)3800

関数ポケットコンピュータ
PC-U6000
全国大学生協連/シャープ

PC-U6000

全国大学生協連とシャープは，関数ポケットコンピュータ「PC-U6000」を全国大学生協連より発売した。

付属機能として，プログラムエディットがしやすいスクリーンエディタ機能や，公式や定数を合計99個まで記憶できる数式記憶機能，240×32ドットサイズで関数，統計，回帰グラフを描画してくれるグラフ機能などを装備。構造型BASICでのプログラミング環境，ユーザーが効率的に利用できるようなサポートがされている。

メインメモリは64KバイトRAM（ユーザーエリア61Kバイト）を標準装備。最大128Kバイトまで拡張ができる。

また，ユーザーが作成したオリジナルソフトウェアを本体に内蔵。収録プログラム内容は，複素数電卓や分子量計算，天誅殺調べと相性診断，通信用サンプルプログラム，POCKET INVADERなど多岐にわたっている。そして，それらの利用法を記載したライブラリ集「電言板6」付き。

価格は35,000円（税別）。

〈問い合わせ先〉
シャープ(株)　☎06(621)1221，043(299)8210

電子メモ
PA-720/520/371
シャープ

PA-720

PA-520

PA-371

シャープは電子メモシリーズとして「PA-720」「PA-520」「PA-371」の3機種を発売した。

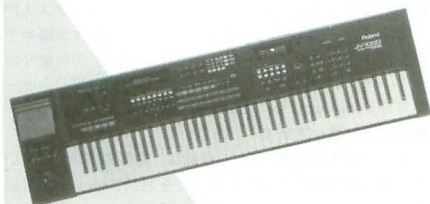
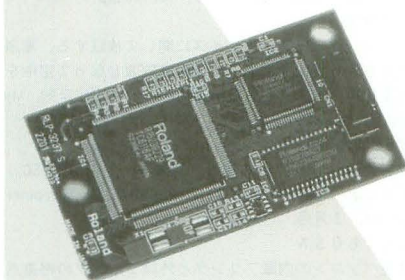
「PA-720」と「PA-520」は3桁表示，「PA-371」は2桁表示の液晶画面をもち，ともに，電話番号とファクシミリ番号など2種類の番号を入力することができる。

また2つの電話帳機能を備え，用途によってビジネス，プライベートなどと使い分けられる。

価格は「PA-720（手帳タイプ）」が5,500円，「PA-520（カードタイプ）」が5,000円，「PA-371（カードタイプ）」が3,800円（すべて税別）となっている。

〈問い合わせ先〉
シャープ(株)　☎06(621)1221，043(299)8210

76鍵ミュージックワークステーション
JV-1000/VE-GS1
ローランド

JV-1000

VE-GS1

ローランドは，76鍵ミュージックワークステーション「JV-1000」を発売する。

「JV-1000」はJV-80をベースにした音源と，MC-80MKII相当のMIDIシーケンサを統合したミュージックワークステーション。

音色はプリセット128+ユーザー64の合計192種類，そして，リバーブ8タイプ，コーラス3タイプのエフェクタを装備するなど，MIDIマスターキーボードとしての使用に耐えられるものになっている。さらに，シンセサイザ部とシーケンサ部を独立させるなど，ミュージックワークステーションにありがちな，操作の複雑さを軽減することを目指している。

内蔵のフロッピーディスクでは，S-MARCで作成したデータの利用，SMFディスクの読み込み，演奏もできる。

また，別売りのボイスエクスパンションボード「VE-GS1」を使うことにより，GS音源がアドオンされ，接続時にはJV音源とGS音源それぞれ28音，最大同時発音数合計56音となる。さらに，GMスコアの読み込み演奏ができるなど，GS標準音楽システムを構築できる。

「JV-1000」「VE-GS1」ともに価格は未定である。

〈問い合わせ先〉

ローランド㈱　☎03(3251)5595

ハーフラックマルチエフェクタ
SE-70
ローランド

SE-70

ローランドはハーフラックサイズのマルチエフェクタ「SE-70」を発売する。

「SE-70」は，35エフェクト，45アルゴリズムを搭載し，ステレオコーラスやボコーダも2タイプ用意されるなど，ハーフラックサイズとしてもかなりの機能を標準装備したマルチエフェクタである。

ギターのディストーション系のエフェクタにはアナログ回路を使用するなど，楽器用エフェクタとしての性能を追求した設計となっている。

また，最大16種類のエフェクタを同時に使用可能。複数エフェクトを自由に組み合わせることで，より幅広い音作りができる。

「SE-70」の価格は未定。

〈問い合わせ先〉

ローランド㈱　☎03(3251)5595

INFORMATION

クローズドキャプションデコーダ CCD-20無料貸し出しサービス実施
日本アイエムアイ

日本アイエムアイでは，テレビとビデオやLDプレイヤーの間に接続するだけで，クローズドキャプション（英語字幕）入りビデオや，LDソフトを再生することのできるデコーダ「CCD-20」の無料試用貸し出しサービスを実施した。

今回のサービスは，CCD-20の普及に伴い消費者からの試用希望の増加に対応するために実施されたもので，希望者にソフト1本付きで2週間の試用を行えるようにしたものである。

貸し出し申込方法は，官製ハガキに無料貸し出し希望，氏名(フリガナ)，住所，電話番号（FAX番号），勤務先（所属学会），希望ビデオソフト名（雨に唄えば，燃えよドラゴン，カサブランカ，バルジ大作戦のうちいずれか1本）を明記のうえ，下記の住所に送付する。

また，官製ハガキだけでなく，電話，FAXでの申し込みも受け付けている。

〈問い合わせ先〉

〒229　神奈川県相模原市矢部1-7-18

日本アイエムアイ㈱　AV事業本部

☎0427(76)9611，FAX0427(76)8772

高速道路コンピュータ・グラフィックスコンテスト
新国土形成研究会

新国土形成研究会では，未来の都市と都市の間を結ぶ高速道路と，新しい日本国土のイメージをテーマとした「高速道路コンピュータ・グラフィックスコンテスト」を開催する。

募集作品は，テーマおよびコンテスト開催趣旨に沿った内容で作成したCGの静止画で，作品形態はカラーポジフィルム（35mm)か，カラーおよびモノクロのグラフィックプリンタなどによる出力原図(A3判以下）に限られる。

応募者はプロ，アマ，職業，国籍，作品製作に使用するハードウェアについても機種を問わない。そして，個人にかぎらず企業単位での応募も可能で，複数作品の応募もできるが，受賞作品はひとり（1団体）1作品となる。

本コンテストでは，最優秀賞1点（副賞20万円），優秀賞2点（副賞10万円），佳作5点（副賞2万円）が用意されている。

なお，審査にあたっては作成技術だけでなく，テーマを的確に捉えているもの，着眼点に優れているものも評価の対象とされる。

応募用紙は，返信用封筒を同封のうえ下記の住所に請求すること。応募期限は平成5年4月30日（金）必着で，審査発表は入賞者に直接通知するとともに，PIXEL10月号でも発表が行われる。

〈問い合わせ先〉

〒990　山形県山形市松波2-8-1　山形県土木管理課内　新国土形成研究会

〝高速道路CGコンテスト〟事務局

☎0236(30)2436

FILES Oh!X

このインデックスは，タイトル，注記――著者名，誌名，月号，ページで構成されています。ゲームはやや低調の今月ですが，新製品X68030の登場の速報が各誌に掲載されています。春の訪れとともにやってきた「新人」はどんなヤツでしょうか？

参考文献
I/O　工学社
ASCII　アスキー
コンプティーク　角川書店
C Magazine　ソフトバンク
テクノポリス　徳間書店
電撃王　主婦の友社
POPCOM　小学館
マイコンBASIC Magazine　電波新聞社
My Computer Magazine　電波新聞社
LOGIN　アスキー

一般

▶MACWORLD EXPOSITION & WINTER CESレポート
アメリカで開催の2つのイベント，Macintoshの展示会「MACWORLD EXPOSITION」と家電製品発表会「CES」からアメリカのゲーム＆マルチメディア業界をみる。――編集部，コンプティーク，3月号，122-124pp.

▶THE NEWS FILE
パナソニックのCD-ROMベースの家庭用ゲーム機「3DO」や，東京・千駄ヶ谷のハイテクショールーム「ヴァンテ」などの話題。――編集部，LOGIN，3号，28-35pp.

▶MUSIC LABO
昨年11月に行われた「マルチメディア'92 MIDI FAIR TOKYO」の模様を通じて各社のMIDI製品などを紹介。――編集部，LOGIN，3号，258-265pp.

▶THE NEWS FILE
IBM互換機への対抗で話題のPC-9800シリーズの最新機種や，ゲームもできるパイオニアのLD-ROM「LaserActive」，東京・王子の日本最大級のゲームセンター「ZYX(ジークス)」など。――編集部，LOGIN，4号，28-35pp.

▶がんばれ！ RPG
パソコンゲームの華「RPG(ロールプレイングゲーム)」について，ソフトメーカーとメインスタッフにアンケート。クリエイターの座談会やユーザーの意見も紹介。――編集部，LOGIN，4号，179-197pp.

▶NEWS CLIP
LD-ROMマシン「LaserActive」，巨大ゲーセン「ZYX」，東京・原宿のデジタルアートギャラリー「DIGITALOGUE」などの話題。――編集部，POPCOM，3月号，26-30pp.

▶新鮮良品館
シャープの小型電子手帳「PA-X2」などの新製品情報。――編集部，POPCOM，3月号，134-135pp.

▶'93 ベストバイ！　パソコン・システム
秋葉原の有名販売店のアドバイスをもとに紹介。X68000XVIや「オーバーテイク」「MATIER」など。――編集部，マイコンBASIC Magazine，3月号，50-57pp.

▶新製品Flash NEWS
X68000用日本語ワープロ「雷語(サンダーワード)」やシャープの電子手帳用周辺機器などを紹介。――編集部，マイコンBASIC Magazine，3月号，71-75pp.

▶パソコン・キャプテンを使ってみたい……人のために
「おさらい編」として，必要なものなどを紹介。――編集部，マイコンBASIC Magazine，3月号，76-77pp.

▶ワープロ/パソコン通信新聞
大手ネットの新サービスや草の根BBSの紹介，初心者向き「パソコン通信への道」など。――山本まさこ，マイコンBASIC Magazine，3月号，78-82pp.

▶Bug太郎のプログラム・タイム　その3
「プログラム実践篇　その1」。シューティングゲームなどに使う，弾の移動方向算出ルーチン。――谷　裕紀彦，マイコンBASIC Magazine，3月号，90-91pp.

▶BASICプログラミング教室　第11回
「確率と乱数でスロット・マシンを作ってみよう！」。サイコロのシミュレーションで確率と乱数を学ぶ。――東幸太，マイコンBASIC Magazine，3月号，92-96pp.

▶ローランドMIDI音源ユーザーのためのGS音源相談室
リバーブとコーラス，定位の変え方について。――後藤浩昭，マイコンBASIC Magazine，3月号，177-189pp.

▶BASIC MAGAZINE NEWSスペシャル
話題のハード3機種を紹介。「LaserActive」「3DO」「PCエンジンDUO-R」。――山下　章，マイコンBASIC Magazine，3月号，160-161pp.

▶山下　章のウィンターCES'93レポート
「CES」で展示されたゲーム関連商品を紹介。――山下章，マイコンBASIC Magazine，3月号，211-225pp.

▶ASCII EXPRESS
X68000の32ビットモデル「X68030」など，パソコン関連最新情報。――編集部，ASCII，3月号，158-177pp.

▶DIGI-VIS TODAY
VTRにおける緻密な合成処理を可能にした業務用デジタル編集機「ハリー」について。――編集部，ASCII，3月号，334-335pp.

▶TBN SPECIAL
「3DO」「LaserActive」などの新世代ゲーム機を紹介。――綾丸，ASCII，3月号，337p.

▶バカパパのモノを買い物
今回は，質感が勝負の金属モノ。ワイヤー加工セット，フロッピーケースなどのグッズ。ほかにホワイトノイズ発生器。――バカパパ，ASCII，3月号，338-339pp.

▶ラッキー！ハッピー！オッケー！
プログラム著作権における私的使用についての法律的解釈の総まとめ。――編集部，ASCII，3月号，360p.

▶だれでもできる実践コンピュータグラフィックス入門
CG特集。TV番組「ウゴウゴ・ルーガ」の制作現場やデジタルプリンタ「ピクトログラフィー3000」の性能リポート，メディアにおけるCGについてなど広範囲に。――編集部，My Computer Magazine，3月号，52-115pp.

▶パソコン買い方教室
「激得！秋葉原攻略法3」。今回は「集客の方法」について。購入客獲得のための宣伝などを解析。――島川言成，My Computer Magazine，3月号，162-163pp.

▶PC実験室
パソコンから出ているノイズに関して検証する。電源コードを通じて出る電動ノイズと電磁波となって空中を伝わる幅射ノイズ，どちらが深刻か？――石川至知，My Computer Magazine，3月号，164-166pp.

▶MYCOM NEWS
システムサコムのX68000対応SCSIボード「SX-68SC」，パイオニア「LaserActive」など。――編集部，My Computer Magazine，3月号，118-204pp.

▶なんでもQ＆A
書院パソコンの内蔵プリンタと外設プリンタの相違点，ハーフサイズのボードの接続方法など。――シャープ，My Computer Magazine，3月号，260-261pp.

▶I/O目安箱
圧縮についての話。圧縮の仕組みとLHAやISHの使い方などを解説。――KANAMI，I/O，3月号，118-122pp.

▶COMPUTER CURRENTS
「初心者のためのマルチメディア」と題し，マルチメディアの世界に入門するためのシステム構築を解説。――ローレンス・J・マギッド，I/O，3月号，145-150pp.

▶スーパーコンピューティング入門26
「カオス」と「フラクタル」の続き。ロジスティク関数グラフを作成。――林智雄，I/O，3月号，151-153pp.

▶New Products
「X68030」など。――編集部，I/O，3月号，161-165pp.

X1/turbo/Z

X1シリーズ

▶AMIDA
線を書いたり消したり…。あみだくじをベースにしたアクションパズルゲーム。――中村　理，マイコンBASIC Magazine，3月号，136-137pp.

▶COIN AND WALL
コインを集めるシンプルなアクションゲーム。――石井秀実，マイコンBASIC Magazine，3月号，138-139pp.

▶ザ・スーパー忍　～China Town～
セガのゲームミュージックプログラム。FM音源ボードとNEW FM音源ドライバが必要。――ひとちゃん，マイコンBASIC Magazine，3月号，155-156pp.

X1turboシリーズ

▶LET'S PROGRAM
素数を求めるプログラムの投稿発表と添削。X1turbo上のBASIC，C，FORTRAN，FORTHのプログラム紹介。――藤本　健，My Computer Magazine，3月号，241-247pp.

X68000

▶GAMING WORLD
シムシリーズ第3弾，アリを飼育(？)するシミュレーション「シムアント」。激辛シューティング「究極タイガー」。ほかに「メガロマニア」「チェルノブ」など。――編集部，テクノポリス，3月号，16-34pp.

▶爆発　攻略野郎
各機種のゲームを攻略。X68000は「信長の野望 覇王伝」。――編集部，テクノポリス，3月号，58-61pp.

▶GAME PARADISE
「チェルノブ」。──編集部，電撃王，３月号，100p.
▶パソコンゲーム新作一覧
２月８日〜３月７日に発売予定のゲームソフトをリストアップ。──編集部，電撃王，３月号，152-153pp.
▶SOFT EXPRESS
「KU²フロントロー」。機種別のNew Soft Indexも。──編集部，コンプティーク，３月号，38，43pp.
▶New Soft Making
移植中のアリ社会のシミュレーション「シムアント」。──編集部，コンプティーク，３月号，52-57pp.
▶最新ゲーム徹底解剖!
「三國志Ⅲ」の官渡，赤壁，五丈原などの地形を解説。多彩なマップが楽しめる「エトワールプリンセス」は，コミカルなキャラクターの紹介とゲームの概要。──編集部，LOGIN，３号，146-149，190-191pp.
▶X68000新聞
X68000に接続できる大容量外部記憶メディアを紹介。ほかに「幻影都市」。「Ｃ言語講座・連載９回」は四則演算を解説。──編集部，LOGIN，３号，266-269pp.
▶SUPER NEW SOFT
「シムアント」。──編集部，LOGIN，４号，18-19pp.
▶最新ゲーム徹底解剖!
「三國志Ⅲ」は一騎討ちについて。武将同士一対一で戦ってみよう。「素人のための三國志Ⅲ」は素人対マニアのマルチプレイ。──編集部，LOGIN，４号，116-119pp.
▶X68000新聞
新作情報は「チェルノブ」「Traum」「雷語（サンダーワード）」「Communication SX-68K」。Ｃ言語講座ではエスケープシーケンスを使って文字色を変えてみる。──編集部，LOGIN，４号，206-207pp.
▶Software Hot Press
開発中の「シムアント」と新作「チェルノブ」。──編集部，POPCOM，３月号，15，22pp.
▶THE 対決！
ズームの「オーバーテイク」を編集部のスタッフ同士で対戦だ！──編集部，POPCOM，３月号，58-61pp.
▶新製品　Flash NEWS
新作ソフトやハードを紹介。「雷語（サンダーワード）」。──編集部，マイコンBASIC Magazine，３月号，74p.
▶Space Defencer
照準を合わせて敵ミサイルを撃破。SDI風アクションシューティングゲーム。ジョイスティック専用。──林　純一，マイコンBASIC Magazine，３月号，140-141pp.
▶PINBALL IN THE SPACE
風でボールを操る障害物ピンボール風ゲーム。──SHMSOFT，マイコンBASIC Magazine，３月号，142-143pp.
▶MAGIC board
じゃまな壁を矢で破壊しながらマジックボードを集める。タイム制限のあるアクションパズルゲーム。──濱口和彦，マイコンBASIC Magazine，３月号，144-146pp.
▶ファイナルファンタジーⅤ　〜オープニング〜
スクウェアのスーパーファミコン用ゲームのミュージックプログラム。要NAGDRV（GS音源）。──Rabettch，マイコンBASIC Magazine，３月号，157-159pp.
▶AV STRASSE
「雷語」，SX-WINDOW上の通信ソフト「Communication SX-68K」，付録ディスクに収録の高機能ファイラ「SuperTF」など。──編集部，ASCII，３月号，285-288pp.
▶TBN GAME
イマジニアの「シムアント」を紹介。──上野利幸，ASCII，３月号，328-329pp.
▶FREE SOFTWARE INDEX
主要BBSにアップロードされたソフトから。X68000用ファイルシステムチェッカやファイルダンプユーティリティなど。──編集部，ASCII，３月号，373-379pp.
▶X68030とSX-WINDOW3.0
「X68030」の詳細をレポート。速度性能，拡張機能からソフトウェア強化部分，将来性についてなど。──高橋雄一，My Computer Magazine，３月号，118-123pp.
▶SOUND SX-68K
サウンドツール「SOUND SX-68K」を紹介。──都築敏也，My Computer Magazine，３月号，232-234pp.
▶なんでもＱ＆Ａ
「Multiword」のバージョンアップ点と「Communication SX-68K」のログのゴミへの対応。──シャープAVCシステム事業推進室，My Computer Magazine，３月号，258-259pp.
▶SLG Laboratory
「飛びモノ」といわれるフライトシミュレータ特集。X68000用「スターウォーズ」も登場。──猪野清秀，My Computer Magazine，３月号，294-299pp.
▶特集　ゲームプログラミング大研究
BASICやＣ言語などを使ったゲーム作成の手法など。X68000用にはX-BASICによる迷路データ展開配置プログラムを紹介。──土方嘉徳ほか，I/O，３月号，17-79pp.
▶GAME BOX
アリのシミュレーション「シムアント」。──鮫嶋直樹，I/O，３月号，86-87pp.
▶GCCで学ぶX68ゲームプログラミング　第16回
今月は，BG画面を使ったラスタースクロールに挑戦。──吉野智興，C Magazine，３月号，126-130pp.

ポケコン

PC-E500
▶レーザー・ショット
レーザーの方向をミラーで曲げるアクションゲーム。──TAKA，マイコンBASIC Magazine，３月号，148p.

新刊書案内

電脳騒乱節VOL.４
オブジェクト指向
臨死体験編
中村正三郎著
技術評論社刊
☎03(3225)2300
A5判　268ページ
1,200円（税込）

電脳騒乱節も最終回を迎えた。昔，はじめて電脳騒乱節で私の名前が出てきたとき（いい意味で登場するわけがない），（で）が「電脳騒乱節に名前が出るなんて有名になりましたね」なんていいやがった。ええい，クソ。結局，都合３回ほど槍玉に上がったわけだが（このVol.４では「読みが浅く筋が悪い」なんて書かれてる），ショックだったのは最終回。この間違いを中村氏から直接指摘されたとき，そんな馬鹿な間違いをするやつがいるか，なんてのけ反ったのだが，自分でやってた。意見やら，主張やら，勘違いを批判されるのなら納得いくが，わかっていてなおかつ惚けてたか熱があったかしてやってしまったアホな間違いを指摘されると，非常につらい。なんであんな間違いしたんだろうなあ。「ああ，恥ずかしい」。

Vol.４では後半，電脳筒井線ネタが急激に出現する。私が中村氏を知るようになったのもこの電脳筒井線であって，人生は不思議だ。

それはともかく，電脳騒乱節Vol.４である。1991年４月から92年３月までの１年分と単行本で追加した解説兼後日談がついている。時事ネタが多い本なので，後日談は必須なのだ。

電脳騒乱節の面白さは，込められた遊びとコンピュータと別の世界を結びつける博学さにある。言葉遊びから論理の遊びからいろんな遊びを織り込んで読者をひきつける。一見コンピュータとは関係ない話とコンピュータを結びつけて話の幅を広げる。個人的に面白かったのは第47回の前半。ネットワーク量子論のあたり。でもって，本音っぽさ，わかんないことはわかんないということで，読者は親近感を持ち，中村氏と同じレベルでモノを見ているような錯覚を持つのである。それでもって日経MIXである。凄いわ。

逆に，こういう遊びや突っ込みを嫌う人もいるわけで，差別的な過激さがある分，反発も多いわけで，そのあたり，確信犯的である。（荻窪　圭）

PC-PAGE32
プログラミング
自由学校
安藤　聡編集
翔泳社刊
☎03(5411)3020
A5判　207ページ
1,600円（税込）

プログラミング言語のマニュアル本ではない。入り口でウロウロしている人，挫折したことがある人など，「プログラミング未満」の人のためのイントロダクションである。理論ではなく，実際的な（しかし，単にアプリケーションソフトを使うだけではない使い方をするための）コンピュータ入門書といってもよい。

本書によると，「切実な問題である」「不便だと思う」「つまらないと思う」「覗いてみたい」というのが「プログラミングのネタの見つけ方４本柱」だそうである。つまり，当然のことながら重要なのは，目的や問題意識だということだろう。

社会・人文系のための
あいまいとファジイ
石川　昭編
オーム社刊
☎03(3233)0641
A5判　240ページ
3,900円（税込）

「ファジィ」という言葉は一般の人々のあいだにも定着したが，概念のほうはといえば，まだまだ「ファジィ」（あいまい）のままのようだ。

しかし，専門家による研究は着々と進んでいるようで，日本ファジィ学会のなかにはノンエンジニアリング（非理工学）ファジィ研究部会も結成されたという。理論としてだけではなく，新しい発想の方法や視点の置き方という点においても，今後さまざまな分野の発展に大きく寄与するであろうということだろう。そういった，政治や経済，心理，社会学などの各分野での現在の取り組み方や応用などが紹介されているのが本書である。

バックナンバー案内

BACK ISSUES

ここには1992年4月号から1993年3月号までをご紹介しました。現在1991年1，5，9，11，12，1992年1，6～12，1993年1～3月号の在庫がございます。バックナンバーおよび定期購読の申し込み方法については170ページを参照してください。

1992

4月号（品切れ）
特集　成熟するゲームと日本の文化

連載　よい子のSX-WINDOW/Z80's Bar
響子 in CGわ～るど/ショートプロ/吾輩はX68000である
ハード工作/ANOTHER CG WORLD/Computer Music入門

●発表　1991年度GAME OF THE YEAR
●バーコードバトラー
LIVE in '92　あじさいのうた/ショパン練習曲作品25-2へ短調/IT'S MAGIC
THE SOFTOUCH　ファーストクィーンII/マスターオブモンスターズII 他
全機種共通システム　実践Small-C講座(1)オプティマイザO80

5月号（品切れ）
特集　明日のための環境づくり
第7回　言わせてくれなくちゃだワ

連載　響子 in CGわ～るど/大人のためのX68000/Z80's Bar
ハード工作/ショートプロ/マシン語プログラミング
Computer Music入門/吾輩はX68000である

●製品紹介　MIDI音源 03R/W/MIC68K
LIVE in '92　フレンズ/Danger Line
THE SOFTOUCH　エイリアンシンドローム/苦胃頭捕物帳 他
全機種共通システム　実践Small-C講座(2)COMMAND.OBJ

6月号
特別企画　Oh!MZ,Oh!X10年間の歩み
特別付録　創刊10周年記念PRO-68K(5″2HD)

連載　響子 in CGわ～るど/大人のためのX68000/マシン語プログラミング
ハード工作/ショートプロ/ANOTHER CG WORLD/Z80's Bar
吾輩はX68000である/Computer Music入門

●新製品紹介　Z'sSTAFF PRO-68K ver.3.0
LIVE in '92　Shake the Street/Ancient relics
THE SOFTOUCH　スピンディジーII/ロイヤルブラッド/ライフ&デス 他
全機種共通システム　実践Small-C講座(3)COMMAND.OBJ2

7月号
特集　超空間美術論
特別付録　DōGA CGAシステム&お試しディスク(5″2HD)

連載　よいこのSX-WINDOW/響子 in CGわ～るど/Z80's Bar
ANOTHER CG WORLD/大人のためのX68000
Computer Music入門/ハード工作/ショートプロ

●試用レポート　V70アクセラレータボード
LIVE in '92　Bye Bye My Love/MATERIAL GIRL/ヴェクザシオン
THE SOFTOUCH　将棋聖天&棋太平68K/シムアース/太閤立志伝
全機種共通システム　実践Small-C講座(4)関数リファレンス

8月号
特集　プログラミング再入門

連載　響子 in CGわ～るど/吾輩はX68000である/よいこのSX-WINDOW
マシン語プログラミング/ハード工作/ANOTHER CG WORLD
大人のためのX68000/Computer Music入門/ショートプロ

●新製品紹介　MATIER/TG100/SOUND SX-68K
LIVE in '92　氷穴/ガラガラヘビがやってくる/風の贈り物
THE SOFTOUCH　三國志III/シムアース/ウルティマVI/バトルテック
全機種共通システム　実践Small-C講座(5)ワイルドカード
グラフィックライブラリGRAPH.LIB

9月号
特集　数値演算の熱い逆襲

連載　DōGA CGアニメーション講座/大人のためのX68000
響子 in CGわ～るど/吾輩はX68000である/ショートプロ
マシン語プログラミング/ハード工作/ANOTHER CG WORLD

●新製品紹介　MATIER/MIREGE Model Stuff
LIVE in '92　恋をしようよ Yeah! Yeah!/ゆめいっぱい
THE SOFTOUCH　ファイナルファイト/ライジングサン/
ヨーロッパ戦線/シューティング68K GAMES
全機種共通システム　O-EDIT & MODCNV

10月号
特集　DTMへの招待

連載　DōGA CGアニメーション講座/大人のためのX68000
響子 in CGわ～るど/吾輩はX68000である/ショートプロ
マシン語プログラミング/ハード工作/ANOTHER CG WORLD

●試用レポート　X68000用CD-ROMドライブ
LIVE in '92　美少女戦士セーラームーン/笑顔を探して 他
THE SOFTOUCH　ポピュラスII/リーディングカンパニー/
ネクタリス/サークII
全機種共通システム　実践Small-C講座(6)SLENDER HUL

11月号
特集　ゲームマネージメント

連載　DōGA CGアニメーション講座/大人のためのX68000
響子 in CGわ～るど/ショートプロ/よいこのSX-WINDOW
ハード工作/ANOTHER CG WORLD/Computer Music入門

●新製品紹介　CHART PRO-68K
LIVE in '92　ストリートファイターII/スーパーマリオ 他
THE SOFTOUCH　キャッスルズ/シュートレンジ/
ポピュラスII/サンダーレスキュー
全機種共通システム　実践Small-C講座(7)EDIT

12月号
Oh!X5周年特別企画　ショートプロ大集合

連載　DōGA CGアニメーション講座/マシン語プログラミング
響子 in CGわ～るど/ショートプロ/よいこのSX-WINDOW
大人のためのX68000/ハード工作/Computer Music入門

●エレクトロニクスショウ'92
LIVE in '92　LAST CHRISTMAS/闇の血族/ユーフォリー
THE SOFTOUCH　デスブレイド/ムーンクレスタ&テラクレスタ/
ふしぎの海のナディア/ロードス島戦記II 他
全機種共通システム　実践Small-C講座(8)MAKE

1993

1月号
特集　D.I.Y.ハードウェア

連載　DōGA CGアニメーション講座/マシン語プログラミング
響子 in CGわ～るど/ショートプロ/よいこのSX-WINDOW
大人のためのX68000/ハード工作/Computer Music入門

●新製品紹介　サンダーワード/SX広辞苑
LIVE in '93　ムーンライト伝説/チャコの海岸物語
THE SOFTOUCH　オーバーテイク/ストライダー飛竜/
エアーマネジメント/パイプドリーム 他
全機種共通システム　実践Small-C講座(9)EDC-Tの拡張

2月号
特集　画像創造のために

連載　DōGA CGアニメーション講座/マシン語プログラミング
響子 in CGわ～るど/ショートプロ/よいこのSX-WINDOW
ハード工作/吾輩はX68000である/Computer Music入門

●新製品紹介　Communication SX-68K
LIVE in '93　FIRE CRACKER/サンバDEグワッシャ!
THE SOFTOUCH　極/ドラゴンスレイヤー英雄伝説/
機甲装神ヴァルカイザー/キングス・ダンジョン
全機種共通システム　BLACK JACK

3月号
特集　X-BASICを学ぶ

連載　DōGA CGアニメーション講座/マシン語プログラミング
響子 in CGわ～るど/ANOTHER CG WORLD/ハード工作
ショートプロ/Computer Music入門/Z80's Bar

●緊急速報　32ビットマシンX68030
●新製品紹介　音源モジュールSC-33/GS音源搭載JW-50
LIVE in '93　ストリートファイターII/晴れたらいいね 他
THE SOFTOUCH　究極タイガー/チェルノブ/シムアント 他
全機種共通　シューティングゲームコアシステム作成法(1)

愛読者プレゼント

カプコン
☎03(3340)0718

ストライダー飛竜

X68000用　5″2HD版

9,800円(税別)　3名

ハイパースタントアクションと呼ばれるとおり，飛竜は地を駆け，宙を舞い，敵を斬る。独特のアクションと見どころ満載の面構成は見事だが，目を奪われて，死んでしまわないように注意しよう。

ファミリーソフト
☎03(3924)5727

スクウェア・リゾート ハイパー戦車戦

X68000用　5″2HD版

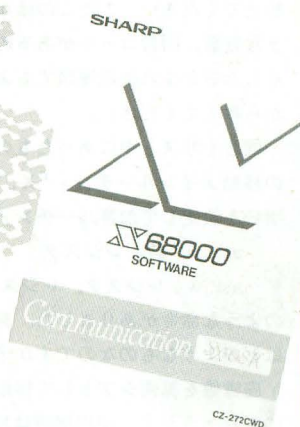
4,500円(税別)　3名

独特の操作性をもつ戦車を操り，敵の戦車と撃ち合うアクションゲーム。とっつきはちょっと悪いけれど，操作に慣れれば結構ハマる。特に2人対戦モードは面白い。画面と音もなかなかのものだ。

シャープ
☎03(3260)1161

Communication SX-68K

X68000用　3.5/5″2HD版

19,800円(税別)　3名

SX-WINDOW用アプリケーションに通信ソフトが登場。特に凝った機能はないが，基本的なところは押さえている。これでSX-WINDOWを起動している時間がさらに多くなるかな。

ソフトバンク
☎03(5488)1360

バーチャル・リアリティ

3,200円(税別)　5名

マルチメディア同様，バーチャルリアリティという言葉は一般に浸透したものの，かならずしも理解されてはいない。この本はその「バーチャル・リアリティ」という言葉をずばりタイトルにし，さまざまな面から光を当てている。

●プレゼントの応募方法●

とじ込みのアンケートはがきの該当項目をすべてご記入のうえ，希望するプレゼント番号をはがき右下のスペースにひとつ記入してお申し込みください。締め切りは1993年4月18日の到着分までとします。当選者の発表は1993年6月号で行います。また，雑誌公正競争規約の定めにより，当選された方はこの号のほかの懸賞には当選できない場合がありますのでご了承ください。

2月号プレゼント当選者

1 ロードス島戦記Ⅱ　(愛知県)船越由裕　(奈良県)片山明義　(大阪府)志摩憲　2 XIN/OUTⅡ ver.7.0f　(栃木県)山野上敬裕　(兵庫県)林秀彦　3 上昇気流 vol.4　(新潟県)広井誠　(栃木県)木村直也　(長野県)宮下誠　(富山県)松本勝正　(福井県)大久保敏之　(岐阜県)今村雄治　(滋賀県)永井徹　(大阪府)松永貴輝　(兵庫県)村瀬正美　(香川県)井筒稔　4 A イマジニアカレンダー　(福島県)高田英夫　(東京都)菊池久男　(愛知県)木下義崇　(宮崎県)岡田晃一　B ソフトバンクカレンダー　(北海道)佐々木淳一　出林聖悟　長谷川聡　(青森県)楝方正治　(栃木県)中村光夫　(埼玉県)小宮崇　筒井圭一朗　(東京都)工藤秀行　(千葉県)五十嵐豊　板垣央　小海崇史　昼間敦史　(神奈川県)臼井悟士　(愛知県)安尾文教　(京都府)小坂友裕　(大阪府)黒田隆之　福森淳　(福岡県)上田考一　(熊本県)岩本信一　境英利　(敬称略)
以上の方々が当選しました。商品は順次発送いたしますが，入荷状況などにより遅れる場合もあります。

QUESTION and ANSWER

1991年10月号特集「マシン語との邂逅」のなかの実践アセンブラプログラミングのコーナーで，浜崎氏の作った星を動かすプログラムリスト中で，

```
21 : FPACK    macro callno
22 :                dc.w   callno
23 :                endm
24 :
25 : _RAND    equ     $fe0e
```

というような，いままで見たことがない命令がありました。

おそらく乱数の宣言，関数のコール文だと思いますが，詳しい使い方や上記5行のそれぞれの意味などがよくわかりません。教えてください。またこのほかにもこのような宣言，関数コールがあるのですか？　もしあるとすれば数種類でもよろしいですから教えてください。

同じくリスト中にあったものですが「星の移動メインルーチン」「X，Y座標から増分を決定して加算ルーチン」のなかに，

```
asr.w  #?, レジスタ
asr.w  レジスタ, レジスタ
```

のような命令がありました。算術シフトとはどのようなものなのですか？　なぜX，Y座標値を算術シフトして移動量を出せるのでしょうか？　使用機種はX68000XVI，HAS.X，HLK.Xです。

東京都　小松　隆夫

最初の質問から回答しましょう。リスト中21～23行はマクロの定義です。まずマクロについて簡単に説明しましょう。

プログラムを組んでいると似たような処理が繰り返し必要になってくることも珍しくありませんね。使用頻度の高い処理はサブルーチンとしてメインルーチンから独立させるのが常です。決まりきった処理といえばX68000のアセンブラでプログラムを組んでいくと，少なからずDOSコールやIOCSコールの呼び出しがあります。ですがIOCSコールの呼び出し処理をサブルーチンにするという人はまずいないでしょう。だって，trap #15自体がサブルーチンコールみたいなものですから。

例として画面モードの変更の場合のIOCSコールの記述を示します。

```
move.w    #12,d1
moveq.l   #16,d0
trap      #15     ＊画面モード変更
```

とはいえd0レジスタに機能番号を設定してtrap #15を実行する手順は決まりきったものです。AS.XやHAS.Xでは，このように決まりきった処理の塊に名前をつけて定義することができます。これがマクロと呼ばれるものです。

マクロは決まりきった手順を単にまとめるだけのものではありません。マクロにはあたかも関数を呼び出すかのように引数も与えることができます。21行の，

```
FPACK     macro callno
```

は，FPACKという名前（Floating-point Packageの略か）のマクロがcallnoという引数を持つことを宣言します。たとえばソースリストに，

```
FPACK     $ff00
```

とあれば，callnoに$ff00が代入されます。ですから22行の，

```
dc.w      callno
```

は，

```
dc.w      $ff00
```

と置き換えられます。23行のendmはマクロ定義の終了を表します。

このマクロは174行あたりで，

```
FPACK     _RAND
```

として使われています。引数は _RANDです。 _RANDは25行で$fe0eと定義されています。ですからFPACK　_RANDは，

```
dc.w      $fe0e
```

となることはおわかりですね。

なお，$fe0eは浮動小数点演算ドライバFLOAT?.Xに組み込まれている乱数発生ルーチンを呼び出す命令です。これらの浮動小数点演算パッケージで追加されるファンクションコールはC compilerPRO-68Kなどの一般のマニュアルには記載されていません。詳しく知りたい場合は数値演算プロセッサボードに付属のマニュアルまたは，工学社刊の『X68000環境ハンドブック』を参照してください。

マクロはソースリストの記述を「わかりやすく」「きれいに」することができますが，その場で展開されるものなので，マクロをサブルーチンコールのつもりで使用していると実行ファイルをいたずらに大きくしてしまいます。

次に算術シフトでなぜ移動量が計算できるか？　ということですが，掲載されたプログラムでの星の移動ルーチンを考えてみてください。記事には「星の表示されているX，Y座標を算術シフトしたものを移動量とする」とあります。これはどういうことかというと，表示されている星の座標位置によって移動量を決定するということです（説明になっていないか？）。リストの一部を抜粋して簡単に説明しましょう。

（抜粋）

```
197: hdx_cal:
198:      moveq.l   #5,d1
199:      tst.w     flag_2(a1)
200:      beq       hd_2
201:      moveq.l   #6,d1
202: hd_2:
203:      move.w    h_x(a1),d0＊星X座標
204:      sub.w     #255,d0
205:      asr.w     d1,d0
206:      add.w     d0,h_x(a1)
```

まずflag_2(a1)の値によってシフト数を決定します。星の明るさによってシフト数は5か6になります。flag_2(a1)が0なら5ですね。次にX座標を取り出し（203行）画面上で0～511の範囲で表している星の表示座標を，プログラム内部で－255～＋256に変換します(204行)。こうして求めたシフト数だけ座標を右にシフトします（205行）右シフトですから，算術結果は2の－5乗，もしくは2の－6乗になります。その演算結果を移動量として元のX座標に足して，新しい表示座標とします（206行）。

これでわかると思いますが，星の明るさによって移動量は2の－5乗（1/32），もしくは2の－6乗（1/64）になります。これで星の明るさによって星の移動量が変わってくるのがよくわかりましたね？　また星が画面の中心座標（255，255）から遠くなるにつれて移動量が大きくなることもわかりますね？（実際に紙に座標を書いてやってみてください）。

このリストでは画面の中心から星が遠くなるにつれて星の移動を速くして画面に奥行きを持たせようとしているわけです。算術シフトで移動量を求められるのは，星の座標位置によって移動量が小さくなったり，大きくなったりするのを利用しているのです。

試しに198行，201行の#5と#6を#3と#6のように，いろいろ数字を変えて実行してみてください。どういう感じになるかわかりますか？

Oh!Xをいつも楽しく読ませてもらっています。ところで僕は，1991年の12月号から買い続けているわけですが，「APIC.FNC」（1992年6月号のディスクに入っていた）の使い方がよくわかりません。記事にはIMG_SAVE，IMG_LOADと同じと書いてありましたが……。お手数ですが，使い方を教えてくれませんか？　東京都　五十嵐　正治

APIC.FNCはAPIC形式でセーブされたグラフィックをロードしたり，その形式でセーブするための外部関数です。BASIC.CNFでAPIC.FNCを組み込んだX-BASICには，

APIC_SAVE
APIC_LOAD

という，以上の2命令が新たに拡張されます。

APIC_SAVEは16，256，65536色（正式には32768色）のグラフィック画像をAPIC形式で圧縮して保存する命令です。保存する矩形領域を長方形の左上点，右下点で指定します。たとえば，

APIC_SAVE("A:¥TEST.PIC", 0, 0, 511, 511)

のように書きます。

APIC_LOADはPIC，APIC形式で保存された画像をグラフィック画面に表示するための命令です。表示する画像の矩形領域の左上点を指定します。

APIC_LOAD("A:¥TEST.PIC", 0, 0)

はAドライブにあるTEST.PICというファイルを座標（0，0）の表示領域の左上点に指定しています。誤動作防止のため必ず表示する左上点を指定してください。これは特にCコンパイラでコンパイルするときに問題になります。基本的な使用方法はIMG関数とあわせてあります。使い方がIMG_LOADなどと違うのは，ロード先頭座標を省略できないことくらいでしょう。

APIC.FNCは私の解析不足とちょっとした勘違いがあって，セーブ時に拡張子PICを自動添付してくれなかったり，ロード時に左上点を必ず指定しないといけなかったりと，自分自身納得いかないプログラムです。ちゃんと組み直せばこういった制限はなくなるはずですので，申しわけありませんが気にいらない方はソースリストを自分でいじるか，次回配布の機会をお待ちください。

なおAPIC形式とはPICで512×512，65536色（32768色）の画像しか保存できなかったものを，16色，256色の画像の保存，再生ができるようにPIC.RをTONBE氏が拡張した圧縮形式です。APIC形式はPIC形式の上位互換となっていますので，APIC形式の画像を表示できるプログラムでは，PIC形式で保存した画像を表示することもできます。逆にPIC.RなどPIC形式で保存した画像を再生するプログラムでAPIC形式で拡張された16色や256色の画面モードで保存した画像を再生することはできません。

RolandのSC-55のエクスクルーシブメッセージでチェックサムというのがありますが，マニュアルを見ても「チェックサムはアドレス，データ，およびチェックサム自身を加算した値の下位7ビットがゼロになる値になっています」と書いてあるだけなのでさっぱりわかりません。システムリセットを例に挙げて説明してください。

栃木県　斉藤　司

X-BASIC風の式の記述で表現するとRoland方式のチェックサムの算出式は以下のようになっています。

SUM＝$80－(アドレス＋データの各バイトの総和) and $7F

ではご要望どおりシステムリセットコマンドを送信する際に必要となるチェックサムを求めてみましょう。

まず，SC-55でシステムリセットを行う場合，全体のエクスクルーシブメッセージ自体は，

F0	エクスクルーシブヘッダ
41	Roland ID
10	デバイスID
42	モデルID
12	コマンドID
40	アドレス
00	（アドレス）
7F	（アドレス）
00	データ
SUM	チェックサム
F7	エンドオブエクスクルーシブ

といった構成になります。

ここで，アドレスは$40007F，データは$00ですから，これをバイトに分割して先ほどの式に当てはめてやります。すると，

SUM＝$80－($40＋$00＋$7F＋$00) and $7F

となりますね。

これを計算して算出される答え，つまり$41をチェックサムとしてSUMの部分に入れて送出すればいいことになります。

なお，この計算方法はSC-55に限らず，Roland製のMIDI楽器すべてに通用するものですのでCM-32/64ユーザーの方も覚えておいてください。

＊　＊　＊

最後に少しばかりお願いがあります。質問箱宛に葉書を送ってくださる方のなかに，ごく一部ですが往復葉書や返信用の封筒を同封してくる方がいますが，都合上，返信にはいっさい応じていませんので，そのような行為はご遠慮願います。なかには現金を送ってくる方までいたりして，そこまで質問箱を頼ってくれるのは担当者として大変嬉しいのですが，現金を送り返す手間もかかりますし，またそれによって優先的に回答するということはありません。今後そのようなことはないようにお願いします。

私が質問箱で回答するのは今月が最後になると思います。約3年間にもわたる長いあいだご愛読ありがとうございました。担当者は変わりますが，来月以降も悩みごとがありましたら質問箱宛にどしどし質問を送ってくださるようお願い申し上げます。それでは，さようなら。　（影山　裕昭）

質問にお答えします

日ごろ疑問に思っていること，どんなことでも結構です。どんどんお便りください。難問，奇問，編集室が総力を挙げてお答えいたします。ただし，お寄せいただいているものの中には，マニュアルを読めばすぐに解答が得られるようなものも多々あります。最低限，マニュアルは熟読しておきましょう。質問はなるべく具体的に機種名，システム構成，必要なら図も入れてこと細かに書いてください。また，返信用切手同封の質問をよく受けますが，原則として，質問には本誌上でお答えすることになっていますのでご了承ください。なお，質問の内容について，直接問い合わせることもありますので電話番号も明記してくださいね。

宛先：〒108　東京都港区高輪2-19-13
NS高輪ビル
ソフトバンク株式会社出版部
Oh!X編集部「Oh!X質問箱」係

FROM READERS TO THE EDITOR

4月です。出会いの季節です。いままでの自分を一枚だけ脱ぎ捨てて，新しい服に着替えませんか。行けなかったところに一歩だけ踏み出してはみませんか。あなたのまわりの新しい風景のなかにも，すてきな何かが待っているはずですよ。

◆最近，ゲームをやらずにプログラムを勉強することのほうが増えてきた。いいことだ。
宮島　雅史(18)長野県

ゲームをやらずにゲームのプログラミングをしているとか……。ゲームでもそれ以外でも，いいのができたら投稿してね。

◆最近のOh!Xの表紙って，過激なのが多いなあ。どうしてでしょうかね。榎本　和則(17)大阪府

◆「GAME OF THE YEAR」を見て思ったのだが，年に1回，「Oh!Xの表紙の絵OF THE YEAR」みたいなのをやってみたらどうだろうか。私は1992年9月号に1票。　谷口　浩史(19)北海道

部門賞として「エキサイティング賞」とか「キャラクター賞」とか，ね。ノミネート作品は，毎年厳選された12点！（あたりまえか）

◆丹さんの記事(2月号特集86ページ)を興味深く読ませていただきました。2次元上での膜の再現についてですが，丹さんは膜の弾力として「曲率保存」を入れておられましたが，これはいんちきではなく的を射ていると思います(生体膜の理論で)。この記事と似たようなシミュレーションが最近行われており，その曲率による弾性エネルギーEは，

$E=K\sum_{i}(1-\cos\Delta a_i)$　(iはノード番号)で表されます。　森河　良太(26)神奈川県

◆2月号の特集は，脳をビシビシと刺激した。「モデル化」という言葉の大切さを改めて考えさせられた。コンピュータの利点は，やはり「シミュレート」にあると思う。こういった知的な楽しさを追求する方向性を今後ももち続けていってください。　野村　慎一郎(19)滋賀県

◆2月号の「CAD_CNV.BAS」はよかったです。私は，Z'sトリフォニーを使っていますが，DōGAへのコンバートを考えていた矢先ですので，タイムリーでした。同じ資源を違うソフトが共有できるというのは，現在のコンピュータ社会でとても重要なことだと思います。こういった思想は未来のマルチメディアにつながるのではないでしょうか。　瀬志本　勇二(20)北海道

好奇心や興味のおもむくままに，いろいろなことを追求していきたいものですね。記事を読んでそこから何かを汲み取り，何かつぎの新しいものを作り出すことにつなげてもらえたら，嬉しいのですが。

◆2月号の「LIVE in '93」の「FIRE CRACKER」は，やっぱり爆竹とかかんしゃく玉って意味だと思う。　宮本　憲和(17)福井県

◆「FIRE CRACKER」ですが，これは「爆竹」という意味のはずです。このことは，YMOのファーストアルバム「YMO」を聴くと明らかです。「FIRECRACKER」は2曲目ですが，1曲目の「サーカスのテーマ」(昔のコンピュータゲームの音のアレンジ)から次第に爆竹の音が混じってきます。ほとんど爆竹の音しか聴こえなくなったところで，「FIRE CRACKER」が始まります。また，曲の終わりにも爆竹の音が鳴っています。だから絶対に「爆竹」です。斎藤　法男(22)滋賀県

そうです，正しい意味は「爆竹」です。斎藤さんの解説どおり，アルバムとしての演出がされているので，1曲だけじゃなくてぜひアルバムで聴いてみてほしい曲ですよね。ところで，2月号の68ページに掲載したジャケット写真は米国版のものです。日本のオリジナル版はどんなジャケット写真だったかなぁ

◆2月号のSOFTWARE INFORMATIONにあった「グラマーな幼児体型」って……僕にはよくわからない……。　河野　裕文(17)静岡県

ではでは「そのミチ20年」(うそぴょん)のワタクシがお教えいたしましょう。一説によると，ロ○コン系(えっち)マンガの女の子の特徴は，「瞳が大きい」「胸が大きい」のほかに「おしりが小さくて足は結構太い」だそうで，ヒップよりフトモモが太いってのがヨイらしい。まあ，件のパッケージを見ていただければ一目瞭然。そのテのものとみまごうばかりの可愛いヤツです。ナンでそういうのにしたのかは謎ですが。ま，可愛ければいーじゃん，ってのかなぁ。

◆2月号のSTUDIO Xを読んでいくと「任意抽出500人で2人のときは200人の読者がいる」とありました。ということは，Oh!Xの読者は約5万人，X68000ユーザーは10万人くらいだから，まだ2倍くらい販売できそうですね。
守田　孝雄(28)京都府

1匹見かけたら30匹……ってのはちょっとちがうか。ところで，Oh!Xの読者のなかにはX68000ユーザーでない人もいるし，逆に複数台所有者もいるから，Oh!Xの部数をあとどのくらい増やせるかというと……。う〜ん，この「難問」は読者に任せて，私はハガキでも読もっと。

◆(浦)さんへ。私は高校時代に開封後3週間の牛乳を飲んだことがあります。普通，口にふくむと甘いはずが少し苦いので，飲み終えてパックを見ると……製造から1カ月！　家族は牛乳を飲まないので，最後に飲んだ覚えがあるのは3週間前。現在，家の戸棚のなかには86年もののミカンの缶詰が……。どうしましょうか？
岩堀　茂生(25)愛知県

(浦)氏の腹痛については，原因が「どれ」だったのかが現在の論争の焦点になっております。本人はゼリー犯人説を唱えてますが，同じもの食べてなんともなかった(ふ)さんはオレンジジュースを主張して譲りま

◀岡村　直也　兵庫県
さあさあ，今月の岡村姉弟ですよ。新機種を前に姉ちゃんの幸せそーな顔！　今年はこのような光景があちこちでくりひろげられるかな。

◀後藤　浩一　大阪府
こちらも新しいマシンの仲間入りのお話ですが……。コレクターのご主人だったら，きっといつまでも大切にしてくれるから，幸せだよね。

▼武田　正道　兵庫県
う〜ん、パワフル。冬の間はさぞかしうずうずしてたんでしょうね。アクティブなひとにはサイコーの季節。テニスにバイク、楽しみ楽しみ。

▼鈴川　美佳子　東京都
楽しい楽しい春だけど、冬のインフルエンザに続いて、花粉症の人には試練のとき。必須アイテムにはくまさんとうさぎさんがポイントよねっ。

せん(彼女はジュースは飲まなかった)。ここに不屈の胃腸の持ち主の岩堀さんが参加してたら……。真実は藪の中！

◆今年はインフルエンザの当たり年だそうで，うちでもまず母がやられ，さらに父がやられ，ついには妹までもがやられてしまいました。そんななかで私だけがひとりピンピンしてるというのは，ひょっとするとX68000ライフの効能でしょうか？　それとも単に私がバ……。
松本　拓司(18)埼玉県

何を食べても腹をこわさない○○さんってもしかしてバ……(でも風邪はひいたらしい)。X68000ライフを満喫しているはかのみなさんはどーなんでしょ？

◆2月号はなんといってもU氏の「microOdyssey」にかぎります。「目標は，手加減しない，他人のことは考えない」これは，命を賭けなければいえないセリフですよ。思わず武者震いをしてしまった。今年1年のOh!Xは期待大です。
細野　純也(20)埼玉県

編集スタッフは毎月「命を賭けて」Oh!Xを作っています。……ってのはちょっと誇大広告かな？

◆ときには立ち止まって，いままで歩いてきた道を振り返ってみることも大切だと思う。その意味で32ビット機への気運が高まる現在こそ，一度いままでの6年間を振り返ってみたい。僕らがX68000とともに追い求め続けているのは夢である。多くの夢が現れ，現実となり，また多くの夢を生んだ。そしてこれからも多くの夢が現れては現実となるだろう。たとえ，プラットホームが変わっても，求めるものは変わらない。着実に歩いていければそれでいいと思う。夢を生み出すことのできるパソコン。僕は次世代にそれを求めたい。　来島　克樹(20)広島県

パソコンの可能性って無限なもののような気がします。それはつまり，使う人間のもっている能力や可能性が無限大だからです。歩く道筋が途中で変わったとしても，夢に向かって歩き続けたいですね。

◆教育学のレポートを書こうとしたときに，ASK68Kは「教育学」を「教育が苦」と変換した。おれの辞書らしい(笑)。
荒田　圭哉(18)神奈川県

「教育額」でも「教育楽」でもなくって，「教育が苦」だったのねぇ。持ち主と以心伝心だったりして。ちなみに担当者のASKでは「教育学」だったけど，X68000用じゃない某FEPで出てきた「教育が句」ってのも，なんだか，おまぬけですよね。

◆1月号165ページの言葉に励まされて，今年の正月に好きな女の子に年賀状を出してしまいました。そしたらなんと返事が返ってきたではないですか！　まだ「告白」という段階ではないですが，これからどうしたらよいでしょうか。

1)　このままつっきる
2)　名前にうにうにと書く
3)　年賀ハガキを見て勉強をがんばる

天達　雄一(17)京都府

やはり，3)を重点的に攻めながら，1)を実行する。というのが妥当なセンかなぁ。でも，2)をしたらどーなるか，てのも興味津々ではあるのだけど。

◆男には理由がある。Oh!Xを買うのにも理由がある。来月号(3月号)を買う理由……「そりゃあ満開の電子ちゃんの続きが気になるからさ！」。男の理由は清くつつましい。
北久保　晴康(30)千葉県

単に2月号では電子ちゃんが登場しなかったから寂しい……んじゃないの？

◆小和田家の3姉妹のなかでは，礼子さんがいちばんいかすと思う。　　織　学(16)大阪府

うっ，さっそくチェックはいってんのか，こいつわ。仮にも未来の妃殿下のご姉妹に対して何たる不埒な発言！　(実はどっちが礼子さんかわかんなくてコメントできない担当者であったりする)

◆例の「CG STEREOGRAM」によると，交差法のできる人なら生身の双子立体視も可能……と書いてありますが，本物の立体視ということはひょっとしたら超立体が見える！　ということでしょうか。小和田礼子さん，節子さん，モデルになってください。　石田　雄二郎(42)大阪府

「超立体」？　2D→3D→4Dってことで時間を超えた何かが見えてくる……?!
ところで，モデルはやっぱり，きんさんぎんさんじゃなくて，小和田姉妹がいいんですかぁ。このぜいたくモノめ(いやべつにきんさんぎんさんがどーってんじゃありませんが)。

◆容姿が似ていると思考や嗜好が似るものでしょうか。Oh!Xを読み返していると，(で)氏の書かれる内容に共感してしまいます。その昔，人に(で)氏に似ているね，といわれたこともあります。ちなみに私も「しょうゆ」は漢字で書けません。しかし，「ばら」は書ける！
加藤　信之(22)東京都

E・A・ポーの小説のなかに，相手の心を読むのには相手の真似をしてみるとよい，という少年の話が出てきます。思考が似てくるってことでしょうか。ちなみに，顔が似てると声も，ってことが多いのは骨格によって響き具合が決まるからだとか。ホントかどうかは「？」ですが。ところで，(で)氏に似てるアナタって，きっと○★△×■(都合により伏せ字)なひとなのね。

◆ずっと昔，先生に「ゲームなんかやっても得るものはないだろう」といわれて，「あるぜ」と答えられなかった自分を思い出しながら，今年もGAME OF THE YEARに1票を投じる。投じる価値があると思うから。中島　民哉(22)埼玉県

◆「CG STEREOGRAM」を買った。そしたら友達が，「そんなもん買ってどうするんだ」と聞いてきた。意外な質問で驚いたが，なるほど，そういう考え方もある。考えてみると，Oh!Xっていう雑誌は意味のないことを真剣にやっている。そして，それを買っている僕は，無意味なものが好きなんだなあ，と思う。でも，そんなOh!Xと自分が好きだ。　安藤　哲(18)山形県

無意味なものから，意味のあるものを生み出すことが重要なのさ，なあんてね。でもとりあえずは，役に立つかどうかじゃなくて，好きなことを追いかけてみるのもいいと思いますけど。

◆ついに私のOh!Xにも，ふくろとじがついていた。あちゃー。　渡辺　洋平(16)北海道

切れてなかった？　すみません。ところで，切ったらえっちなページが出てくることを期待したりしませんでしたか？

◆Mさんの続報です。歳が離れている(たったの6つなのに)という理由でごめんなさいされてしまいました。ガクッ！　みんな，今日から僕のことを「XVIの寅さん」と呼んでくれ。
北川　亮(22)東京都

う〜ん，もうちょっと歳とったら感じ方も変わってくるかも。16歳と22歳だと離れているって感じもするけど，22と28だったらそうでもないし，72と78だったらもうあんまり違わないよね？　あと何年かしてやっぱり好きだったら，再アタックしてみよう！

◆案の定，仮住まいでは必要最低限のものしか出しちゃダメ，といわれてSC-55は接続して1週間もたたずに箱の中に入ってしまいました。とりあえず，いまは音楽データの打ち込みだけ

をやってます。リスト全部打ち終わったあと，どんな曲が流れてくるのか，ワクワクしながら実行させるのが楽しいっていうのに，つまんない(たいてい打ち間違いですぐには曲は流れないけれど)。春になったらたまった(そうでないと困る)データを聴くぞ。くすん。

碓井　理恵(25)和歌山県

春になって「定住」するのが楽しみですね。我慢のあとに聴く曲はまた格別，よね。

◆久びさに長電話をしてしまった。夜10時半頃から明け方5時までにおよびました。しかも自分で切ったわけではなく，コードレスホンの充電池がなくなったので勝手に切れてしまったため，という……。親にばれないように布団をかぶって寝ながら電話する様子は，さながらイケナイテ○クラを楽しむおにーさんのようでした。なんてマヌケな。それにしても男性って長電話しませんよね。えらい。

鈴川　美佳子(20)東京都

え，男性って長電話しないの？　私の知人(♂ね)で，月の電話代が2～3万円ってヤツがいましたけど……(パソ通してるわけじゃなくって純粋に電話代)。遠距離恋愛してるんでもない。まあ，自分のウチならいいけど，公衆電話の長話は迷惑だから気をつけましょうね(あまりにも待たされてキレそーになったことがある……)。

◆ハガキの年齢の欄を書き込んでいて気がついた。私の年齢を2倍すると68になる。意味はないけど。　石川　栄一(34)新潟県

私の年齢を4倍すると68になります……ってのはサバの読みすぎかぁ。だいたい17歳じゃ酒もタバコもペケペケもだめだしなぁ。意味はないけど。

◆2月号137ページの清水君へ。「ヤダモン」は3月で終わっちゃうんだって……残念だね。シクシク。　奥田　直也(20)愛知県

どこぞに「嘆願書」を出すとか……。

◆釧路沖地震はなかなかすごかったですよ。震度6なんてもちろん初体験でしたからね。でも，うちのある地区は電気，ガスは止まらずにすんだので，海側に住んでいる人に比べるとかなり運がよかったです(でも，部屋の壁紙が2カ所裂けてしまった)。　鈴木　賢吾(22)北海道

けがはありませんでしたか？　ものが壊れたりとかは？　壁紙が2カ所ですんだのだったら，まあ不幸中の幸いなのかも。地震なんて「予防」のしようもないしねぇ。

◆なぜレコード大賞をCD大賞といわないのだろうか？　中津　謙(24)大阪府

名前を変えてしまうと，看板とかいろんな書類とか，いろいろなものを作りかえなくちゃならないとか，いままでの記録をどう扱うかややこしいからいやだとか，いや，もしかしたら単にめんどくさいとか……。いやいや，私たち一般人には窺い知れない，ふか～い事情や哲学的問題が存在しているのかも(はんまかいな)。

◆X68000が出てくるマンガといえば，あの「ああっ女神さま」を描いている人の「逮捕しちゃうぞ」の5巻の58話と7巻の73話です。なんかプログラミングもできる方みたいで，フローチャートが描いてあります。「VSに戻る」と書いてあるのが，時の流れを感じさせます。

堀井　努(18)静岡県

藤島康介さんですか。「逮捕しちゃうぞ」に出てくるとは知りませんでした(といって，本屋に向かわんとしている私……)。

◆僕も西川善司さんと同じくらい「ザナドゥ」が好きでした。どのくらい好きかというと，テープが伸びきってついに読み込めなくなったくらい。でも，いまはユーザーテープを作るのに40分以上も待っていられないだろうな。ファルコムさん，X68000版を作ってください。

田端　秀章(20)北海道

それだけ夢中になれるものに出合えたアナタは幸せ者！　X68000版が出るといいね。

◆B-TRONに続け！「雷語1号」に対抗して私もエディタを考えてみた。イナセな紳士「江戸語1世」(エドワードいっせい)。……しかし，名前ばかりが先走ってしまい，内容が全然決まらない。やはり，名前も中身も作ってしまう祝氏はすごい。　伊藤　直也(22)静岡県

ま，カタチからはいるってのもダイジよね。せっかくのアイデアだし，何年か計画で作ってみちゃうってのはどお？　祝氏に弟子入りするとかしてさぁ。

◆昨年暮れに新しいクラリネットを買いました。木製の楽器というのは使い込むほど音の鳴りがよくなるものなので，これからじっくりと時間をかけて自分の思いどおりの楽器に育てあげていくつもりです(気分はまるで「プリンセスメーカー」)。　松永　正弘(22)京都府

楽器だけじゃなく車なんかもそうだけど，道具って，少しずつきたえて自分好みにしていく楽しみがありますよね。クラリネットだとリード磨きとかの「楽しみ」もあるし……。「ドとレとミの音が出ない」なんてことにならないように，大事にしてね。

◆この間，広島のスキー場に行ったとき，お昼に浅間山荘風の怪しい食堂で「焼きめしカレー」という変わったものがあったので，注文してみると焼きめしにカレーのルー(ボンカレー)をかけただけだった。食べてみると焼きめしの味とカレーの味がお互いにジャマしあって，わけのわからない味であった。やはりスキー場でおいしいものを期待するのは無理なのだろうか。

横田　紀明(25)山口県

「浅間山荘風」ってのはいかにもアヤしいなぁ。で，その「焼きめしカレー」って，いくらだったんですか？

◆2月号の川原です。X68000のおかげで，女の子にモテモテで毎日がハッピーです。ところで何でパソコンで女の子にモテるかというと，あまり詳しくは教えられませんが，昔はゲーセンに行っても，女の子はあまりいませんでした。しかし最近は，UFOキャッチャーなどのせいもあって女の子はたくさんいます。僕はそこに目をつけたのです。ひょっとしたら，この子たちは面白いゲームを求めているのではないかと。ここでX68000の出番です。画面も音楽もいい(SC-55があればもっといい)X68000で，女の子はいちころです。口説き方については，また来月にでも。では。　川原　啓(18)群馬県

そうか，スポットはゲーセンだったのか。で，かんじんの口説き方はまだおあずけなのね。はやく，教えてぇ。

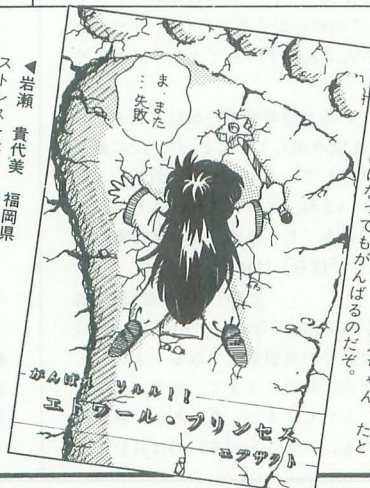
◆12月号の私のつまらない質問について，さまざまな方からいろいろなお答えをいただきまして，本当にありがとうございます。これで，夜眠れない日々が終わることでしょう。あとは，焼き豚とチャーシューの区別ができれば，悪夢にうなされることもなくなるでしょう。

横田　耕一(22)愛知県

あのぅ，ここはお料理教室ぢゃあないのですが……。チャーシュー麺にはいってるのは，チャーシュー！　あとは……？　違いってあるのかな。

◆ニワトリですか。アメリカでも鳴き方は同じで(実際に聞いたわけではない)アメリカ人にはそう聞こえるらしいですね。おまけに向こうの人には，秋の虫の声も雑音にしか聞こえないそうです。　小林　勝(24)奈良県

右脳で聴くか，左脳で聴くか，ってことですよね。でも，アメリカ育ちの日本人とか，

◀岩瀬　貴代美　福岡県
ストレスをぶつけられるゲームがない，とお嘆きの岩瀬さんですが……。そのエネルギーをイラスト投稿にぶつけるってのは，いかがかな？

◀中村　正人　新潟県
ああぁ。い，痛そう……。「どど～ん」ってすごい音が聞こえてくるみたい。リルルちゃん，たとえ傷だらけになってもがんばるのだぞ。

日本生まれのアメリカ人とか，混血の人はどうなんでしょうね。それから，やっぱり個人差もいくらかあるのかなぁ。それがわかれば，悪夢にうなされることもなくなる……はっ，いかん！　このままではSTUDIO Xが「なんでも相談室」に……。

◆つい最近，耳たぶのところに直径3mmくらいの小さなしこりができてしまいました。なんでしょう，これ。私の友達にも最近できました。宇宙人のしわざだったりして。

熊谷　育哉(19)愛知県

なんかアヤシイことして，アヤシイびょーきを流行らせているのでは？　そんなこといっているアナタが実は宇宙人だったりして……。そのしこりは，母船からの司令を受け取るための特殊受信機でしょ？　ほらほら白状しなさい。いまなら特別に許してあげましょう。皇太子さまも結婚なさることだし……。

◆STUDIO Xにイラストなどを投稿するときのルールがよくわかりません。どこかに「ハガキでもいいけど，裏に消印を押されちゃうときが……」とあったように思いますが，封書でもいいんでしょうか。それとサイズは？　教えてください。

大谷　育子(24)大阪府

はっきりしたルールがあるわけじゃないけど，いちおーの目安をお答えしましょう。

・ハガキサイズ　　少しくらいなら違ってもいいけど，あんまり形の違うもの(細長いヤツとか)はたぶん載りません。
・ハガキでも封書でも可　　封書のほうが汚れたりする危険は少ないかも。ハガキでもまず大丈夫ですが。イラストの裏にも住所や名前を書いてくださいね。あと，コメントとかも書いていいですよ。
・鉛筆描きは載りません。
・基本的には白黒ね。

とゆーことで，投稿大歓迎！　だよん。

▲玉野　健一　奈良県
花キャベツみたいな帽子が可愛い！　ちょうちょが寄ってきたりして。この本が発売される頃には危篤のディスプレイは元気になってるかな？

ぼくらの掲示板

●掲載ご希望の方は，官製ハガキに項目(売る・買う・氏名・年齢・連絡方法……)を明記してお申し込みください。
●ソフトの売買，交換については，いっさい掲載できません。
●取り引きについては当編集部では責任を負いかねます。
●応募者多数の場合，掲載できないこともあります。
●紹介を希望されるサークルは必ず会誌の見本を送ってください。

売ります

★X1turbo用データレコーダ「CZ-8RL1」を10,000円前後，立体映像セット「CZ-8BR1」10,000円で売ります。両方とも箱はありませんが，付属品，マニュアルはすべてあります。連絡は往復ハガキでお願いします。〒230　神奈川県横浜市鶴見区朝日町2-89-312　村中　隆志(22)

★X1/X1turbo用５インチフロッピーディスクドライブ「CZ-503F」を8,000円，X1用14型カラーディスプレイ「CZ-811DE」(説明書のみあり)を12,000円，X1/X68000用24ピンカラー漢字プリンタ「CZ-8PC2」(カラーリボン６本付き)を18,000円，PC-TV352を35,000円で売ります。「CZ-811DE」以外は付属品一式あり，「CZ-811DE」は箱がないため手渡しを希望します。連絡は官製ハガキでお願いします。〒167 東京都杉並区上井草2-26-9　秋山　和徳(21)

★後期型Roland製MIDI音源モジュール「MT-32」，カラーイメージユニット「CZ-6VT1」を各20,000円で売ります(送料込み)。ともに説明書と付属品はありますが，箱はありません。官製ハガキで連絡してください。〒223 神奈川県横浜市港北区日吉2-26-32　松浦　信生(20)

★X1/X68000用24ピン漢字プリンタ「CZ-8PC1」を送料込み5,000円で売ります。第２水準漢字ROM，ケーブル，箱，マニュアル，インクリボン付きです。連絡は往復ハガキでお願いします。〒244　神奈川県横浜市戸塚区矢部町353-15-402　山岸　茂樹(36)

★X68000用サイバースティック「CZ-8NJ2」を5,000円以上で売ります。付属品，箱すべてあり。往復ハガキで連絡してください。〒389-08　長野県更級郡上山田町温泉2-25-7　山崎　高志(19)

★X1用カラーイメージボード「CZ-8BV1」を送料込み7,000円で売ります。箱，付属品すべてあり。連絡は往復ハガキでお願いします。〒231 神奈川県横浜市中区北方町1-58　山崎　正富(50)

★X1/X68000用24ピン熱転写プリンタ「CZ-8PC2」を10,000円(箱，取扱説明書あり)，X1用FM音源ボード「CZ-8BS1」を5,000円で売ります(ともに送料別)。連絡は必ず往復ハガキでお願いします。〒334　埼玉県鳩ケ谷市里153　高野　宏(21)

★X68000用カラーイメージユニット「CZ-6VT1」を25,000円で売ります。送料別で着払いで送付します。連絡は往復ハガキでお願いします。〒079-12　北海道赤平市平岸新光町5-2　北条　彰也

★X68000用カラーイメージユニット「CZ-6VT1」を送料込み36,000円で売ります。箱，付属品あり。連絡は往復ハガキでお願いします。〒272 千葉県市川市大野町1-82-3　石原　大助(17)

★Roland製MIDI音源モジュール「CM-32L」＋X68000用MIDIボード「CZ-6BM1A」＋ミキサー「BX-4」をセットで50,000円で売ります。箱，付属品，マニュアルなどすべてあります。連絡は往復ハガキでお願いします。〒704 岡山県岡山市西大寺浜150-2　吉川　貢司(19)

★X1用データレコーダ「CZ-8RL1」を8,000円，X68000用カラープリンタ「CZ-8PG1」を40,000円(付属品，箱なし)，X1/X68000用24ピン漢字プリンタ「CZ-8PK5」を20,000円で売ります。連絡は往復ハガキで。〒940　新潟県長岡市東神田3-4-12　伊藤　宏幸

買います

★X1用５インチディスクドライブユニット「CZ-503F」または同等の性能をもつ単体の完動品を8,000円前後，X1turbo用カセットデータレコーダ「CZ-8RL1」を6,000円，「CZ-8BR1」の付属ソフトなしを10,000円以下で買います。連絡は往復ハガキでお願いします。〒230 神奈川県横浜市鶴見区朝日町2-89-312　村中　隆志(22)

★X68000用2Mバイト増設メモリを30,000円前後で買います。箱，付属品，取扱説明書ありを希望します。希望金額を書いて往復ハガキで連絡してください。〒618　大阪府三島郡島本町山崎1-8-39　信垣　直嗣(19)

★X68000用2Mバイト増設メモリボードを20,000円で買います。拡張スロットに差し込むタイプであれば，メーカーを問いません。連絡は往復ハガキでお願いします。〒328　栃木県栃木市片柳町2-53-7　山野上　敬裕(19)

バックナンバー

★Oh!X1988年2,3,5,6月号を送料込み各1,000円で買います。４冊まとめてなら5,000円で買います。切り抜き不可。連絡は往復ハガキでお願いします。〒253　神奈川県茅ケ崎市菱沼1-6-17　野田　敏之(21)

編集室から from E・D・I・T・O・R

DRIVE ON

このコーナーでは，本誌年間モニタの方々の意見を紹介しています。今月は２月号の内容に関するレポートです。

●特集を読んで思い出しましたが，私が小学生のとき，図画の時間に木の絵を描かされたことがありましたが，そのとき木の葉がいろいろな緑色に見えて，とても色を塗るのに手間どっていました。苦労をして仕上げても，ちっとも本物らしく見えなかったものでした。コンピュータなら，そんな自然さを簡単に出してくれるものかと思いましたが，やはり，いまも昔も自分ががんばらなくてはならないことに変わりはないのですね。

金井　徳之(20)　X68000 ACE　千葉県

●特集にあった「AMIGAのScenary Animator& VISA PRO」ですが，ちゃんと使えるソフトとして，上記のようなものが出回っていることに「やっぱりAMIGAだな」と感心させられてしまいました。記事自体もレビューとしてかなりまとまりがあったと思います。また，「X68000次世代へのかけ橋」についてですが，単にスペックだけにとらわれずに，データの互換性や機能の拡張性といったところに目を向けている文章には，とても感動させられました。とにかく私は，カタログスペックばかりを追求してしまいがちなので，基本的な使い方などは二の次くらいにしか思ってなかったように感じます。もともと使っているマシンがX68000なので，こう思ってしまうのかもしれませんが（それだけX68000がオペレートしやすいマシンだということです）。

藤田　康一(22)　X68000 PRO　静岡県

●２月号の特集を読んで，率直にいってしまうと「10年後ならもっと完璧にできたな」という感じがしました。炎とか柔らかいプリミティブとかのテーマは，一個人が動くにはちょっとテーマが重すぎたかな，というような気がします。記事を書いている方もさぞかし欲求不満が溜まったことでしょう。確かに「こういうのがやりたかったんだよ」と嘆きたくなるようなことも，たくさん書かれていました。たぶん，いつかこの記事に感謝する日がくることでしょう。私個人としては木の生成が好きです。あと，２月号では「キングス・ダンジョン」が面白そうだな，と思いました。ふふ～ん，と笑いながら遊べそうなところがいいです(コワイ……)。普段は敵であるモンスターが，ここでは自分。もし，かわいい勇者がいたらどうしよう，おっかない勇者がいたらどうしよう……結局いままでと同じことか。でも，たまには立場が反対になるのもいいかもしれない。

安井　百合江(18)　X68000 PRO　愛知県

●自然の植物をCGで自動生成させようとするには，ランダムに作っていくだけではうまくいかないような気がします。木の生長には，太陽の向きや気温，気圧などの変化がからんでくるでしょうから。あと，現状のままでは何も生まれてこないということです。いくら「MATIER」がすごくても，できないことはあるのです。我々自身がなんとかせねばどうにもならない，そう感じました。２月号の特集で，可能性を示しただけでも意味があったのではないでしょうか。

酒元　一幸(20)　X68000　千葉県

●２月号で「GAME OF THE YEARノミネート発表」が行われましたが，今年は「ファイナルファイト」を始め，X68000らしいゲームが多い年でした。私はそのテのゲームにあまり興味がないため，ちょっと寂しい１年でした。ただ，「三國志Ⅲ」や「大戦略Ⅲ'90」にははまったし，「ロードス島戦記Ⅱ」もなかなか面白かったかな。でも，「バトルテック」は残念でした。全体としては，ありがちなパターンで終わった１年であったと思います。

宍戸　輝光(19)　X68000 PRO, MSX2　東京都

●次期X68000って「ムズカシイキカイ」でないといけないんですか？　色がきれいで音がよい。もちろん，なんにもおまけを付けないでですよ。そして，そこそこがまんのできるディスクアクセスの時間。これだけでよいのではないでしょうか。便利なものをあれこれ付けてくれるのもいいですが，使いにくいのはとっても嫌です。今月号に限らず，私は「micro Odyssey」が好きです。これって，書いている人の人格が出ているように思うんです。人の書いたものは人が読むからいいですよね。コンピュータのように抑揚のないしゃべり方って聞きたくないです。コンピュータっていう得体のしれない変なものを使っている人が，人間くさいことを書くのを読むのが大好きです。

野原　志貴乃(30)　X68000 ACE-HD　埼玉県

●２月号の特集は，いかにも「パワーユーザのためのOh!X」にふさわしい特集だと思いました。が，やはり限られた誌面では，きっちり説明するのは難しかったようですね。こういったものに対する読者の反応は，すっぱり分かれてしまうと思いますが，あえてチャレンジするのはやっぱりOh!Xですね。

三津田　哲雄(18)　X68000 EXPERTⅡ　山口県

ごめんなさいのコーナー

２月号　(で)のショートプロぱーてぃ

P.118　リスト１QREWD.BASで，通常ミス時にゲームオーバー時のPCMが鳴ってしまうようになっていました。リスト１の510行と520行を以下のように書き換えてください。

```
510 if qrd>1 then {m_play(12)}
520 } else {m_stop( ):m_play(13)}
```

また，940，950行でPCMファイル名の指定が間違っていました。

```
940 QRD5.PCM→QRD2.PCM
940 QRD6.PCM→QRD3.PCM
950 QRD4.PCM→QRD1.PCM
```

以上のように変更してください。

３月号　(で)のショートプロぱーてぃ

P.132　リスト３，PCM_MAKE.BATの最初の１行が抜けていました。抜けている１行は以下のとおりです。

```
zvt -c %1 t1.tmp
```

３月号 Oh!X LIVE in '93

P.112　ケンのテーマ（リスト４）の印刷状態が悪く，判読不能な部分がありました。今月号の75ページLIVE in '93に再掲載されていますので，そちらをご参照ください。

バグに関するお問い合わせは
☎03(5488)1311(直通)
月～金曜日16：00～18：00

お問い合わせは原則として，本誌のバグ情報のみに限らせていただきます　入力法，操作法などはマニュアルをよくお読みください　また，よくアドベンチャーゲームの解答を求めるお電話をいただきますが，本誌ではいっさいお答えできません　ご了承ください

新しい風が吹いた我々の進むべき道を探そう

▶今月号では，特集「X68第7世代へ」と題して，新登場のX68030をさらに紹介しています。特徴的なのは，68EC030について多くのページが割かれている点です。もちろん今回のバージョンアップの目玉ともいえるのですが，やはり，使える読者が多くいるOh!Xならではなものでしょう。

そして，このメインMPUである68EC030の概要については，いままで使ったことも見たこともない人にとって，非常に参考になったと思います。記事を読めばわかるとおり，オブジェクトレベルでの互換性を保っている点やアドレッシングモードの追加など，いままで68000を使ってきた人にとっては，まさに魅力あふれたものでしょう。システムやアプリケーションでも，SX-WINDOW ver.3.0，Human68k ver.3.0やASK68K ver.3.0など，変更されたものは多くあります。それらが，いままでのX68000で使用できる点も見逃せません。

X68030の登場でさらに広がりをもったX68ワールドが，どう発展していくか。また，どう発展していくべきか，読者の皆さんも考えてみてください。

▶1992年GAME OF THE YEARの発表が行われました。ここには，読者の皆さんからの投票の結果があります。この結果をただの人気投票で終わらせるのではなく，どんなゲームがどんな理由でウケたのか，いま一度考えてみるのもいいかもしれません。なお，読者レビューは投稿された数が少ないこともあり，該当作品なし，ということにさせていただきました。

▶読者参加の特別企画，「言わせてくれなくちゃだワ」が目前に迫りました。ばしばしアンケートが返送されてきて，いまからどんな意見が飛び出すか楽しみです。また，イラスト関係はまだまだ掲載する数に余裕がありそうなので，がんばって投稿してください。

▶「大人のためのX68000」がまたまたお休みとなってしまいました。来月こそは復活しますので，楽しみにしていてください。

投稿応募要領

●原稿には，住所・氏名・年齢・職業・連絡先電話番号・機種・使用言語・必要な周辺機器・マイコン歴を明記してください。

●プログラムを投稿される方は，詳しい内容の説明，利用法，できればフローチャート，変数表，メモリマップ（マシン語の場合）に，参考文献を明記し，プログラムをセーブしたテープ（ディスケット）を添えてお送りください。また，掲載にあたっては，編集上の都合により加筆修正させていただくことがありますのでご了承ください。

●ハードの製作などを投稿される方は，詳しい内容の説明のほかに回路図，部品表，できれば実体配線図も添えてください。編集室で検討のうえ，製作したハードが必要な場合はご連絡いたします。

●投稿者のモラルとして，他誌との二重投稿，他機種用プログラムを単に移植したものは固くお断りいたします。

あて先

〒108 東京都港区高輪2-19-13 NS高輪ビル
ソフトバンク出版部
Oh!X「㋢㋑㋮㋯」係

S H I F T · B R E A K

▶童謡のしゃぼん玉。きっと誰でも知っているだろう。しかし，その隠された意味は？　実は野口雨情さんの子供が生まれてすぐに亡くなったときに作詞したものであるという。そういえば思い当たる歌詞もある。生まれてすぐに，こわれて消えた。これを知ったときは，まさに目からウロコが落ちる思い。久しぶりに童謡のCDを聴いてしまった。（SIVA）

▶テレビが壊れたとき，どうせならゲームもできたほうがいいやと思いゲームギアを買った。CDラジカセが壊れたときも，同じことを考えてDUOを買った。そして最近どうも洗濯機の調子が悪いのだ。NEO GEOかスーファミに洗濯機がつく予定はないのだろうか？　関係ないが正月に行った熱川のバナナワニ園は衝撃的だった。でももう行かない。（哲）

▶ゲーセンがキレイになって女の子が増えた。だけど先日黄色い声をあげながら6ボタンを叩きまくる女の子がいたのには仰天。あまりのヘタさに指導してあげたけど，まったくの素人がいきなりターボに座って，プレイしながら「昇龍拳はどう出すの？」たってねぇ。君の場合まず防御を覚えたほうが。次のストⅡは女の子向けバージョン？（浦）

▶東京都の三森浩一さん「宇宙クリケット大戦争」の文庫本どうもありがとう。私もいろいろな本屋で探し回ったのに見つからなかったんです。だからうれしいです。三森さんはルディ・ラッカーの「時空の支配者」と「空を飛んだ少年」を探しているそうですが，見かけた人がいたら情報をお知らせください。それにしてもX68030ほしいな。（初代機ユーザー善）

▶今月の愛読書「Think GNU」で，リチャード・ストールマンの「Appleから買うな！」というなかなか刺激的な言葉を見つけた。詳しい経緯は同書を読んでいただくとして，過度の権利の主張がソフト業界を崩壊させるという彼の主張に賛成。真似をして「フジテレビで観るな！」といいたいところだが，いえないのが情けない。代替物がないから。（A.T.）

▶ペアだとかトリプルだとか，あげくは166詰め込みロープウェイなどと大量輸送効率化へまっしぐらな昨今，いちばん隅の14年前に作られた，かつては主役だったはずの長距離リフトに腰掛けると，なにやら気持ちがいい。それは拡声器がないためにうっとおしい音楽がないだけなのだが，一瞬，遊園地からスキー場へワープしたような錯覚を覚えた。（K）

▶セーラームーンが佳境に入ったと聞いたのがきっかけだった。昔はキャンディキャンディ目当てでよく立ち読みをしたものだが，こうしてふたたび手にすることがあるとは夢にも思わなかった。最初は，一度だけと軽い気持ちだったけど，TVのCMを見てついつい買い続けている。恐るべし，講談社。ああ，今月は公式ファンブックが付録か。（KO）

▶待望の春がくる前に，つらい季節がある。気温の変化へのアレルギー。一日中，くしゃみくしゃみくしゃみ。喉の奥がかゆくって，鼻と耳がつながってることを思い知らされる。花粉は関係ないようだし，予防なんてできない。子供のころはなんともなかったのに。人類は進化の過程にあると信じているアタシだけど，個体としては退化してるのかも。（ふ）

▶最近「聖伝」がだれている。たいてい壮大なスケールで描く……などというものは，途中でだれるからしようがないとはいえ，やっぱり長くなるだけ読者を惹きつけ続けるのは難しいんだなあ。なんにしても，停滞している繰り返しだけの状況はつまらないからね。でもまあ，「源氏」がそれなりに動きだして面白くなってきたからいいや。（J）

▶新しいマシンを買ったら，あの最高傑作といわれるF1ゲームが速くなった。スゴイ。CHIP RAMへのアクセスが速くなって，カスタムチップも一新されたからだろう。やっぱりCPUだけ速いなんてのはダサイぞ。“インテル，入ってなーい”と声を大にして叫びたい。かくして，AMIGA1200がサブマシンになり，SE/30はサブ²マシンになった。（A）

▶隆，ブランカ，春麗，ケン，本田，ザンギ，バイソン，サガット，ベガ……。ふぅ，Ⅱ'全キャラ1コインクリアまであと3人。巷はTURBOだってのにねぇ。「なんですか，この火の鳥は？」「どうもフェニックスと呼ばせたがってるみたいですね」じゃあ小さいのはアンドロメダか……と反射的に思ってしまった自分が情けない。（U）

▶予約注文が殺到して生産が追いつかないって？　そりゃそうでしょう。もしかしてシャープさんは自分たちが作ったマシンを過小評価してないかなあ。ともかく，シャープさんは私たちユーザーにできない部分をサポートしてくださいね。何度もいうように問題は周辺環境です。いっぱいあるでしょ。あれとこれとあれとあれと……。（T）

microOdyssey

知れば知るほど使ってみたくなる。それが，X68030に対する感想だ。

もっとも魅力的に感じたのは，680EC30そのものだ。68000のアセンブラに，なによりも愛着を感じている僕のことだから，その仕様を知るたびに，うなったり，ワクワクしたり，何ができるか考え込んだりできて非常に楽しい。

また，新しいシステム，特にSX-WINDOW ver.3.0が快適に遊べるのもいい。

X68030発表から，世間ではさまざまな評価が乱れ飛んでいるが，現在までX68000が作り上げてきたものを思い出していただきたい。過去の資産をそのまま継承でき，より高速，より高性能なMPUで実行できるだけでもかなりの魅力がある。

しかし，この互換性が足を引っ張っているのも事実だ。X68030が中途半端な印象を受けている人も，この「MPU部分が代わっただけ」という仕様が気にかかるのだろう。過去をいっさい捨てて，新しいものを目指そうとしたX68000に魅せられた人にとっては，非常にもの足りないところがある。

そして，僕自身それなりに評価しているX68030だが，いますぐにX68030への移行をするか，というとそうでもない。寒い貯金残高はともかく，ぶっちゃけた話，高速なX68000のイメージから脱却していないからだ。処理速度が上がり，できることをより深く追及できる土壌ができた部分は評価しよう，だが，それ以上のものがまだ見つからないのである。

僕は，仕事場であるOh!X編集部でX68000 XVIを使っているが，動作クロックを16MHzにすることは滅多にない。まあ，僕自身X68000 EXPERTIIを所有していることもあり，家と仕事場での環境が違うのは嫌だ，というのがひとつの理由としてある。しかし，最大の理由は，やりたいことが10MHzでもできる，という現状だ。使いたいものは揃っているし，10MHzでもそれなりに動く，それらを駆使してやりたいことも実現可能だろう。要するに僕は，X68000に対する限界が見えていないのだ（自分自身に対する限界はしょっちゅう感じるが）。そのため，高速化の必要性を感じないのである。

それに，自分の心の中では，技術力が足りないから速いマシンを使いたい，などという理由を許したくない。X68030でしかできないことが見つかったとき，初めてX68030の扉を叩くことになるだろう。

もしも，買い足したとしても，Z80から68000へ移行するまで，多少時間がかかった僕のことだから，実際にX68030を使い始めるまで，しばらくの間X68000を使い続けると思う。融通のきかない頑固者といわれるかもしれないが，実際にZ80で培った経験は68000を使うときに役立ったし，環境が同じであるX68030を使おうとしたときには，いままでやってきたことが何ひとつ無駄にはならないはずだ。

まだ，進んでいくべき道はわからない。問題は山積みだし，それこそ進んでいこうとするその先に何が待っているかわからない。しかし，X68030の登場で大きな1歩を踏み出したことは誰もが認めるだろう。僕たちユーザーがしっかり先を見据える力，マシンに対する思いを忘れなければ道が開けていくのだから。　（J）

1993年5月号4月17日(土)発売

特集　襲撃SX-WINDOW
・SX-WINDOW ver.3.0詳報

第8回　言わせてくれなくちゃだワ

Oh!X LIVE in'93
もう笑うしかない/「弟切草」より「日記帳」
全機種共通システム
シューティングゲームコアシステム作成法(3)

バックナンバー常備店

東京	神保町	三省堂神田本店5F	03(3233)3312
	〃	書泉ブックマートBI	03(3294)0011
	〃	書泉グランデ5F	03(3295)0011
	秋葉原	T-ZONE 7Fブックゾーン	03(3257)2660
	八重洲	八重洲ブックセンター3F	03(3281)1811
	新宿	紀伊国屋書店本店	03(3354)0131
	高田馬場	未来堂書店	03(3209)0656
	渋谷	大盛堂書店	03(3463)0511
	池袋	旭屋書店池袋店	03(3986)0311
	八王子	くまざわ書店八王子本店	0426(25)1201
神奈川	横浜	有隣堂横浜駅西口店	045(311)6265
	〃	有隣堂ルミネ店	045(453)0811
	藤沢	有隣堂藤沢店	0466(26)1411
神奈川	厚木	有隣堂厚木店	0462(23)4111
	平塚	文教堂四の宮店	0463(54)2880
千葉	柏	新星堂カルチェ5	0471(64)8551
	船橋	リブロ船橋店	0474(25)0111
	〃	芳林堂書店津田沼店	0474(78)3737
	千葉	多田屋千葉セントラルプラザ店	0472(24)1333
埼玉	川越	黒田書店	0492(25)3138
	川口	岩渕書店	0482(52)2190
茨城	水戸	川又書店駅前店	0292(31)0102
大阪	北区	旭屋書店本店	06(313)1191
	都島区	駸々堂京橋店	06(353)2413
京都	中京区	オーム社書店	075(221)0280
愛知	名古屋	三省堂名古屋店	052(562)0077
	〃	パソコンΣ上前津店	052(251)8334
	刈谷	三洋堂書店刈谷店	0566(24)1134
長野	飯田	平安堂飯田店	0265(24)4545
北海道	室蘭	室蘭工業大学生協	0143(44)6060

定期購読のお知らせ

Oh!Xの定期購読をご希望の方は綴じ込みの振替用紙の「申込書」欄にある『新規』『継続』のいずれかに○をつけ，必要事項を明記のうえ，郵便局で購読料をお振り込みください。その際渡される半券は領収書になっていますので，大切に保管してください。なお，すでに定期購読をご利用の方には期限終了の少し前にご通知いたします。継続希望の方は，上記と同じ要領でお申し込みください。

海外送付ご希望の方へ

本誌の海外発送代理店，日本IPS(株)にお申し込みください。なお，購読料金は郵送方法，地域によって異なりますので，下記宛必ずお問い合わせください。

日本IPS株式会社
〒101 東京都千代田区飯田橋3-11-6
☎03(3238)0700

Oh!X 4月号
■1993年4月1日発行　定価600円(本体583円)
■発行人　橋本五郎
■編集人　稲葉俊夫
■発売元　ソフトバンク株式会社
■出版事業部　〒108 東京都港区高輪2-19-13　NS高輪ビル
Oh!X編集部　☎03(5488)1309
出版営業部　☎03(5488)1360 FAX 03(5488)1364
広告営業部　☎03(5488)1365
■印　刷　凸版印刷株式会社

満開の電子ちゃん

作・え　岡村祭

日本音楽著作権協会（出）許諾願届出中（ウソ）

講読方法：定期購読もしくはソフトベンダーTAKERUでお買い求めいただけます。
★定期購読の場合＝購読料6ヶ月分6,000円（送料サービス、消費税込）を、現金書留または郵便振替で下記の宛先へお送り下さい。
　現金書留の場合：〒171　東京都豊島区長崎1-28-23 Muse西池袋2F　㈱満開製作所
　郵便振替の場合：東京　5－362847　㈱満開製作所
●ご注文の際は、郵便番号・住所・氏名・電話番号を忘れずに記入して下さい。
●3.5インチディスク版をご希望の方は、「3.5インチ版」とご指定下さい。
●新規購読の方は「新規」と明記して下さい。なお、特に購読開始号のご指定がない場合は既刊の最新号からお送りいたします。
●製品の性格上返品には応じられませんが、お申し出があれば定期購読を解約し残金をお返しします。
★TAKERUでお求めの場合＝1部につき1,200円（消費税込）です。
●定期購読版と内容が一部異なる場合があります。御了承下さい。
●お問い合わせ先　TEL(03)3554－9282（月～金　午前11時～午後6時）
（なお、定期購読版のバックナンバーについては定期購読の方のみご注文を承ります）

志磨　雅則（東京都）

私がX68Kを買って半年ほどたった頃に届いた『電脳倶楽部』のダイレクトメール。これが私の運命を変えたといっても過言ではありません。ブキミな名前で不安だったのですが、いざ届いたものをのぞくとこいつがスゴイ！読者の方の技術の高さはそれまで市販ソフトで満足していた私の認識を一気に突き崩したのです。
読者のハガキに鋭いツッコミをいれる編集部のギャグセンスも並じゃありません。X68Kを、カルトマシーンに格上げ（下げ？）したのも、電脳倶楽部があるからだと私は確信しています。

3月25日発売予定 予価380円(税込)

The スーパーファミコン |スーパーファミコンまるかじり!| 臨時増刊号

シューティング大特集号

★シューティングゲーム・データランド

今までに発売されたシューティングゲームに関するさまざまなデータを集積

★新作シューティングゲーム徹底攻略

スターフォックス(任天堂)

開発者インタビューのほか、スーパーFXチップ、ポリゴンなどの話題も満載

Pop'nツインビー(コナミ)

コナミシューティングの歴史も紹介

バイオメタル(アテナ)

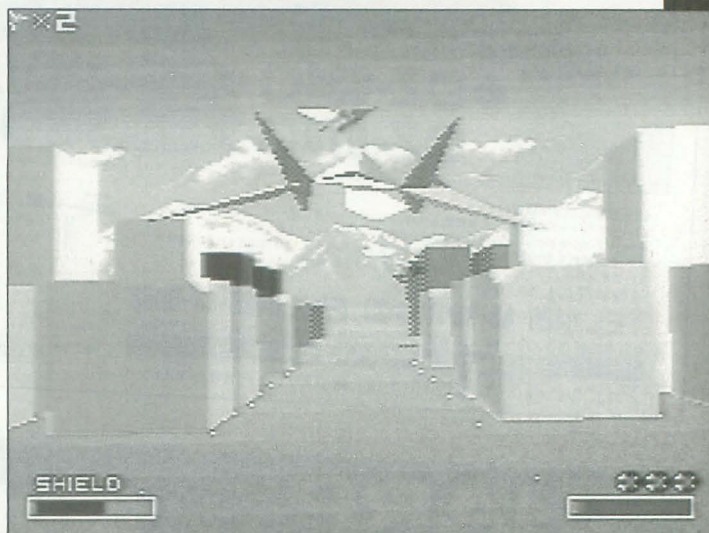
スーパーバトルタンク(パックインビデオ)

デザート・ストライク(EAビクター)

怒りの要塞(ジャレコ)

★最新シューティングゲーム先取り紹介

ダライアスIII(タイトー)

コットン(データムポリスター)

★シューターへの道 シューティングゲームが上達する秘訣は!?

★ボスキャラ大紹介 名作シューティングゲームの名物ボスを一挙大紹介

★シューティングゲームの歴史 アーケードからコンシューマまで詳細に紹介

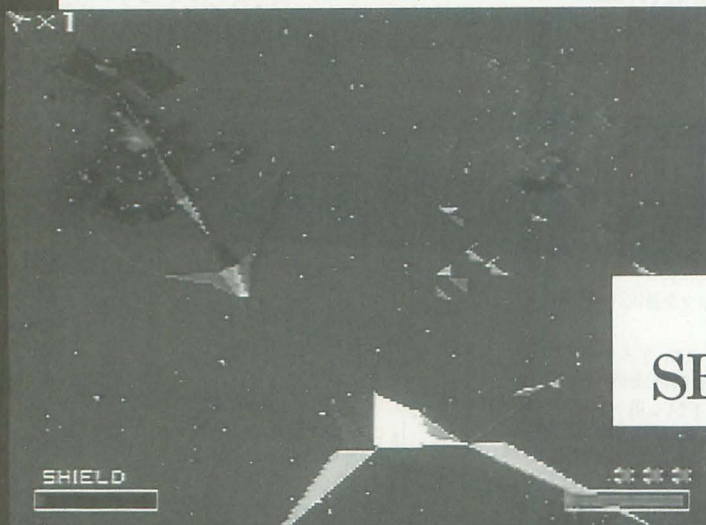
★SFCオールシューティングゲームカタログ

93年2月までに発売されたSFCのシューティングゲームのパーフェクトガイド

★あの名作シューティングゲームをSFCに

アーケードゲームを含む他機種のシューティングゲームがSFCに移植されるかどうかをメーカーに問う。アーケード、スーパーNESの最新情報も網羅。

特 別 付 録

SFCシューティング裏ワザ読本

SOFT BANK

全国の書店、コンビニエンスストアにて発売予定 ソフトバンク出版事業部

《業界No.1の"P&Aメンテナンスサポート》

最高の保証システム

①業界最長の新品パソコン5年保証
（※モニター・プリンター3年間保証.// ※一部商品は除きます。）
②中古パソコンの1年間保証
（モニター・プリンター6ヶ月間保証）
③初期不良交換期間3ヶ月
（※新品商品に限らせていただきます。）
④永久買取保証
⑤配達の指定OK//（土曜・日曜・祭日もOK.//）
⑥夜間配送もOK//
（※PM6:00〜PM8:00の間 ※一部地域は除きます。）

便利でお得な支払いシステム

①翌月一括払い手数料無料（ご利用下さい。）
②業界No.1の低金利
③月々の支払いは￥1,000より
④9ヶ月先からのスキップ払いOK//
⑤84回までの分割、ボーナス併用OK//
⑥カレッジクレジット
⑦ステップアップクレジット
⑧ボーナスだけで10回払いOK//
⑨現金一括払いOK//
⑩商品到着払いOK//（代引き手数料が必要になります。）
（※商品・金額ご確認の上、銀行振込・現金書留にてご入金下さい。）

●法人向けリースシステム
業務に最適なシステムを構築します。
損金処理が可能なリース契約をどうぞ。

注目!!夏のボーナス一括払い手数料(金利)無料
（平成5年4月末／5月末／6月末／7月末のいずれかをご指定下さい。）

でっかく 新装!!
（SHARP＝X68030エキスパートショップ）
P&A 新本店OPEN!!

P&A＝X68030 ダブルNEWフェア
《3月18日〜4月17日》

フェア記念 第3弾

32ビットX68030いよいよ登場。購入ダブルチャンス!!

今だからこそ選ぶ限定セット

32ビット X68030

●CZ-500CB（本体）……定価￥398,000
●CZ-604DB（ディスプレイ）定価￥ 94,800
合計定価￥**492,800**
（送料・消費税別）

P&A特価￥お電話下さい。
（プレゼント＝発売記念、ゲームソフト、ディスケット）

注目!! 限定20台

■CZ-604D
（ブラック）
定価￥94,800

●14″ 0.31mm
●スピーカ、チルトスタンド付

⊙TVチューナー付のモニター（CZ-613Dグレー）に変更の方は￥27,000加算して下さい。
■CZ-613D（グレー）
定価￥135,000
●15″ 0.31mm
●TVチューナー、スピーカー、チルトスタンド付

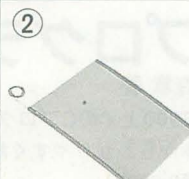
X68030発売記念

X68030をモニターとセットで／単品で）購入の方

さらに現在お持ちのパソコンと下取り交換されたお客様に期間中もれなく、

①サイバーステック（CZ-8NJ2 ￥23,800）
②CRTフィルター（BF-68PRO ￥19,800）
③X-68000フロッピーアタッシュケース（￥8,000）
とクリスタルポルシェ（￥8,000）

以上のいずれかプレゼント!!

① ② ③

●本広告の掲載の商品の価格については、消費税は含まれておりません。

P&A 全国通販

32ビットX68030いよいよ登場（送料¥2,000・消費税別）

①CZ-500CB…定価¥398,000（本体）
CZ-608D(B)･定価¥ 94,800（ディスプレイ）
合計定価¥492,800
▶特価TEL下さい。

②CZ-500CB…定価¥398,000（本体）
CZ-614DTN･定価¥135,000（ディスプレイ）
合計定価¥533,000
▶特価TEL下さい。

①CZ-510CB…定価¥488,000（本体）（80MB HD内蔵）
CZ-608DB…定価¥ 94,800（ディスプレイ）
合計定価¥582,800
▶特価TEL下さい。

②CZ-510CB…定価¥488,000（本体）（80MB HD内蔵）
CZ-614DTN･定価¥135,000（ディスプレイ）
合計定価¥623,000
▶特価TEL下さい。

注目!!夏のボーナス一括払い手数料（金利）無料（平成5年4月末／5月末／6月末／7月末のいずれかをご指定下さい。）

旧シリーズ 今が買いどき!!

X68000 Compact XVI／XVI-HD

送料¥2,000、消費税別（クレジット表：送料、消費税込み）

Compact XVI

① ●CZ-674C-H（本体） ●CZ-608D-H（モニター） ●CZ-6FD5（5" FDD）
定価¥492,600
P&A超特価¥285,000

12回	26,000	24回	13,700	36回	9,500	48回	7,400

上記のモニターをCZ-614Dに変更

② ●CZ-674C-H（本体） ●CZ-614D-TN（モニター） ●CZ-6CR1（RGBケーブル） ●CZ-6CT1（TVコントロール） ●CZ-6FD5（5" FDD）
定価¥542,800
P&A超特価¥318,000

12回	29,000	24回	15,300	36回	10,600	48回	8,300

XVI

① ●CZ-634C-TN（本体） ●CZ-608D-H（モニター）
定価¥462,800
P&A超特価¥249,000

12回	22,700	24回	12,000	36回	8,300	48回	8,400

上記のモニターをCZ-614Dに変更

② ●CZ-634C-TN（本体） ●CZ-614D-TN（モニター）
定価¥503,000
P&A超特価¥260,000

12回	23,700	24回	12,500	36回	8,400	48回	6,800

XVI-HD

① ●CZ-644C-TN（本体） ●CZ-608D-H（モニター）
定価¥612,800
P&A超特価¥389,000

12回	34,400	24回	18,200	36回	12,600	48回	9,900

上記のモニターをCZ-614Dに変更

② ●CZ-644C-TN（本体） ●CZ-614D-TN（モニター）
定価¥653,000
P&A超特価¥415,000

12回	36,700	24回	19,400	36回	13,400	48回	10,500

X68030、X68000をセットで

お買い上げの方にもれなくプレゼント!
①ディスケット10枚、ゲームソフト1ケはもちろん、さらにその上、人気の
㋑オーバーティク（¥9,800） ㋺ロードス島戦記II（¥9,800）
㋩三国志III（¥14,800） ㊁デスブレイド（¥9,800）
㋭エトワールプリンセス（¥9,800）
の中のいずれか1本をプレゼント!!

上記①のモニターを

●CZ-607D（定価¥ 99,800）に変更の場合¥ 3,000
●CU-21HD（定価¥148,000）に変更の場合¥33,000
を加算して下さい。

X68000シリーズ～P&Aスペシャルセット（送料¥2,000・消費税別）

SUPER-HD ★ハードディスク81MB搭載!! ※ディスケット10枚・ゲームソフト1ケプレゼント

Ⓐセット：■CZ-623C-TN(単品)……………定価¥498,000▶特価¥158,000
Ⓑセット：■CZ-623C-TN+CZ-606D………定価¥577,800▶特価¥213,000
Ⓒセット：■CZ-623C-TN+CZ-608D………定価¥592,800▶特価¥226,000
Ⓓセット：■CZ-623C-TN+CZ-607D………定価¥597,800▶特価¥248,000
Ⓔセット：■CZ-623C-TN+CZ-614D………定価¥633,000▶特価¥248,000
Ⓕセット：■CZ-623C-TN+CU-21HD………定価¥646,000▶特価¥258,000

PRO-II P&A特選セット 限定 ※ディスケット10枚・ゲームソフト1ケプレゼント

Ⓐセット：■CZ-653C(単品)……………定価¥285,000▶特価¥129,000
Ⓑセット：■CZ-653C+CZ-606D……………定価¥364,800▶特価¥186,000
Ⓒセット：■CZ-653C+CZ-604D……………定価¥379,800▶特価¥188,000
Ⓓセット：■CZ-653C+CZ-608D……………定価¥379,800▶特価¥198,000
Ⓔセット：■CZ-653C+CZ-607D……………定価¥384,800▶特価¥200,000
Ⓕセット：■CZ-653C+CZ-614D……………定価¥420,000▶特価¥220,000
Ⓖセット：■CZ-653C+CU-21HD……………定価¥433,000▶特価¥230,000

パソコンにワープロがついているユニークな商品

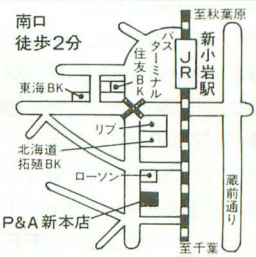
DOS/Vマシン 書院パソコン

⊙PC-WD1A……定価¥330,000
P&A超特価
¥198,000

DOS/V取説ビデオプレゼント!!

〔銀行振込でお申し込みの方〕（電信扱いでお振込み下さい。）
〔振込先〕さくら銀行 新小岩支店
当座預金 2408626
（株）ピー・アンド・エー

超低金利クレジット率

回数	3	6	10	12	15
手数料	2.9	3.9	4.9	5.4	8.4
回数	24	36	48	60	72
手数料	11.4	15.9	20.9	26.9	34.9

P&A
株式会社ピー・アンド・エー
〒124 東京都葛飾区新小岩2丁目2番地20号
☎03-3651-0148(代) FAX.03-3651-0141
●営業時間：AM10:00～PM7:00
日・祭：AM10:00～PM6:00
●定休日/毎週水曜日

●価格は流通事情により変動致しますので、銀行振込・書留等の送付前に、あらかじめお電話にてご確認下さい。

注目!!夏のボーナス一括払い手数料(金利)無料(平成5年4月末/5月末/6月末/7月末のいずれかをご指定下さい。)

P&Aならではの 新品パソコン 5年保証

《業界No.1の"P&Aメンテナンスサポート》

最高の保証システム

①業界最長の新品パソコン5年保証
(※モニター・プリンター3年間保証!! ※一部商品は除きます。)
②中古パソコンの1年間保証
(モニター・プリンター6ヶ月間保証)
③初期不良交換期間3ヶ月
(※新品商品に限らせていただきます)
④永久買取保証
⑤配達の指定OK!!(土曜・日曜・祭日もOK!!)
⑥夜間配送もOK!!
(※PM6：00〜PM8：00の間 ※一部地域は除きます。

便利でお得な支払いシステム

①翌月一括払い手数料無料(ご利用下さい。)
②業界No.1の低金利
③月々の支払いは¥1,000より
④9ヶ月先からのスキップ払いOK!!
⑤84回までの分割、ボーナス併用OK!!
⑥カレッジクレジット
⑦ステップアップクレジット
⑧ボーナスだけで10回払いOK!!
⑨現金一括払いOK!!
⑩商品到着払いOK!!(代引き手数料が必要になります。)

(※商品・金額ご確認の上、銀行振込・現金書留にてご入金下さい。)

モデム(送料¥1,000 消費税別)

■FMMD-311G
(富士通)定価¥35,800
▶特価**¥24,800**
(送料・消費税込み¥26,574)

■PV-M24V5
(AIWA)定価¥36,800
▶特価**¥25,700**
(送料・消費税込み¥27,501)

■MD-24FB5V
(オムロン)定価¥39,800
▶特価**¥23,500**
(送料・消費税込み¥25,235)

●お近くの方は、お立寄下さい。専門係員が説明いたします。
●本体単品でも受付します。詳しくは、お電話にてお問合せ下さい。

●ビジネスソフト定価の15%引きOK!! TEL下さい。

《増設メモリー&数値演算プロセッサ》計測技研 (送料¥500・消費税別)

品名	定価	特価
①PRKII-02(2M)	¥55,000	¥39,800
②PRKII-04(4M)	¥90,000	¥67,000
③PRKII-06(6M)	¥125,000	¥92,500
④PRKII-08(8M)	¥160,000	¥119,000
⑤PRKII-12(2M)	¥85,000	¥63,000
⑥PRKII-14(4M)	¥120,000	¥92,000
⑦PRKII-16(6M)	¥155,000	¥114,500
⑧PRKII-18(8M)	¥190,000	¥146,000
⑨MC-68881RC	¥38,000	¥27,000

FDD(5インチ×2基)
■CZ-6FD5
(シャープ)
(定価¥99,800)
P&A超特価
¥49,800

X68000専用ハードディスク(外付)(送料¥1,000・消費税別)

■ロジテック
⦿LHD-FM100E
●100M ●19ms
定価¥99,800
▶超特価TEL下さい。

■ロジテック
⦿LHD-FM200E
●200M ●17ms
定価¥138,000
▶超特価TEL下さい。

■システムサコム
⦿HD-J130
●130M ●20ms、富士通、純正ドライブ使用
定価¥148,000
▶特価**¥59,500**

■ジェフ
⦿GF-240
●240M ●16ms
定価¥148,000
▶特価**¥95,000**

周辺機器コーナー (送料¥1,000・消費税別)

品名	定価	特価
①CZ-8NSI	¥188,000	¥133,000
②CZ-6VTI	¥69,800	¥49,500
③CZ-6TU	¥33,100	¥23,900
④BF-68PRO	¥19,800	¥14,400
⑤CZ-8NM3	¥9,800	¥7,200
⑥CZ-8NT1	¥13,800	¥10,000
⑦CZ-6BE2A	¥59,800	¥42,800
⑧CZ-6BE2B	¥54,800	¥39,300
⑨CZ-6BE2D	¥54,800	¥39,300
⑩CZ-6BF1	¥49,800	¥35,800
⑪CZ-6BP1	¥79,800	¥57,000
⑫CZ-6BM1	¥26,800	¥19,300
⑬AN-S100	¥36,600	¥26,300
⑭CZ-6SDI	¥44,800	¥32,500
⑮CZ-6BN1	¥29,800	¥21,500
⑯CZ-6BV1	¥21,000	¥15,200
⑰CZ-6BC1	¥79,800	¥57,000
⑱CZ-6BG1	¥59,800	¥43,000
⑳CZ-6BU1	¥39,800	¥28,500
㉑CZ-6PV1	¥198,000	¥142,000
㉒CZ-6BS1	¥29,800	¥21,500
㉓CZ-8NJ2	¥23,800	¥17,500
㉔CZ-6BL2	¥298,000	¥214,000
㉕JX-100S	¥89,800	¥44,000
㉖JX-220X	¥168,000	¥121,000
㉗IO-735XB	¥248,000	¥135,000
㉘LC-10CIH	¥598,000	¥459,000
㉙CZ-6CSI(674C用)	¥12,000	¥8,900
㉚CZ-6CR1(RGBケーブル)	¥4,500	¥3,600
㉛CZ-6CT1(テレビコントロール)	¥5,500	¥4,400
㉜CZ-6BP2	¥45,800	¥33,300

■SX-68MII(MIDI)(サコム)
定価¥19,800(送料・消費税込み¥14,935)
特価**¥13,500**

■CZ-68HA
●674C用内蔵HD80M
特価**¥91,000**

プリンター(ケーブル用紙付 送料¥1,000・消費税別)

■CZ-8PC5-BK
定価¥96,800
▶特価**¥68,500**
■CZ-8PK10
定価¥97,800
▶特価**¥71,000**

カラーイメージジェット

■IO-735X-B
定価¥248,000
特価**¥135,000**
(送料・消費税込み¥140,080)

X68000メモリボード

①SH-6BE1-1M(600C専用)(I/Oデータ)……定価¥25,000
特価**¥17,900**(送料・消費税込み¥18,952)
②1MB増設RAMボード(ACE/PRO/PROII用)…定価¥25,000
特価**¥15,900**(送料・消費税込み¥16,892)
③2MB増設RAMボード(拡張スロット用)……定価¥50,000
特価**¥31,700**(送料・消費税込み¥33,166)
④4MB増設RAMボード(拡張スロット用)……定価¥88,000
特価**¥55,200**(送料・消費税込み¥57,371)

P&A特選パソコンラック (送料無料)

①3段**¥8,900**(消費税込み¥9,167)
●1230(H)×600(D)×650(W)
②4段**¥9,900**(消費税込み¥10,197)
●1250(H)×700(D)×640(W)
③5段**¥12,500**(消費税込み¥12,875)
●1310(H)×700(D)×640(W)

全機種=移動自由(キャスター付)●コードクランプ付(4段/5段)
※5段のみ=電源コード付(2.5m)(2P)キーボード収納可能

OAチェアー

¥11,500(消費税込み¥11,845)

¥19,800(消費税込み¥20,394)

X68000用ソフトコーナー (送料¥700・消費税別)

品名	定価	特価
◆Z's STAFF PRO68K Ver.3.0(ツァイト)	¥58,000	¥37,500
◆Z's TRIPHONY デジタルクラフト(ツァイト)	¥39,800	¥27,000
◆テラッツォ(ハミングバード)	¥19,400	¥13,600
◆マジックパレット(ミュージカルプラン)	¥19,800	¥14,200
◆たーみのる2(SPS)	¥17,800	¥13,000
◆Mu-1 Super	¥39,800	¥28,500
◆CMA68K(シティソフト)	¥29,800	¥21,800
◆サイクロン EXPRESSα68	¥98,000	¥69,000
◆C-TRACE68 Ver.3.0(キャスト)	¥98,000	¥68,500
◆C&Professional Pack V3.2(マイクロウェアジャパン)	¥80,000	¥57,800
◆ウエットペイント1〜3(ウエーブトレイン)〔各〕	¥15,000	¥11,500
◆マチエール(サンワード)	¥39,800	¥28,800
◆Windex PRO68(JEL)	¥28,000	¥20,500
◆CZ-213MSD MUSIC PRO68K	¥18,800	¥13,200
◆CZ-214MSD SOUND PRO68K	¥15,800	¥11,300
◆CZ-215MSD Sampling PRO68K	¥17,800	¥12,500
◆CZ-220BSD DATA PRO68K	¥58,000	¥40,000
◆CZ-224LSD The 福袋 Ver.2.0	¥9,980	¥7,400
◆CZ-225BSD Multiword Ver.1.1	¥32,000	¥23,000
◆CZ-243BSD CYBERNOTE PRO68K	¥19,800	¥15,000
◆CZ-247MSD MUSIC PRO68K (MIDI)	¥28,800	¥20,500
◆CZ-249GSD CANVAS PRO68K	¥29,800	¥22,000
◆CZ-251BSD Hyper word	¥39,800	¥29,400
◆CZ-253BSD CARD PRO68K Ver.2.0	¥29,800	¥22,700
◆CZ-257CSD Communication PRO68K Ver.2.0	¥19,800	¥15,300
◆CZ-258BSD Teleportion PRO68K	¥22,800	¥16,900
◆CZ-261MSD MUSIC studio PRO68K Ver.2.0	¥28,800	¥21,200
◆CZ-263GWD Easypaint SX-68K	¥12,800	¥9,800
◆CZ-265HSD New Print Shop Ver.2.0	¥20,000	¥15,400
◆CZ-266BSD Press Conductor PRO68K	¥28,800	¥22,000
◆CZ-267BSD CHART PRO68K	¥38,000	¥29,800
◆CZ-272CWD Communication SX68K	¥19,800	¥14,500
◆CZ-275MWD SOUND SX68K	¥15,800	¥11,500
◆CZ-284SSD OS-9/X68000 Ver.2.4	¥35,800	¥25,600
◆CZ-285LSD C-Compiler PRO68K Ver.2.1	¥44,800	¥32,500
◆CZ-286BSD BUSINESS PRO68K Popular	¥28,000	¥20,500
◆CZ-287SS SX-WINDOW Ver.2.0	¥12,800	¥9,800

☆ゲームソフト25% OFF OK!!(一部ソフト除く)

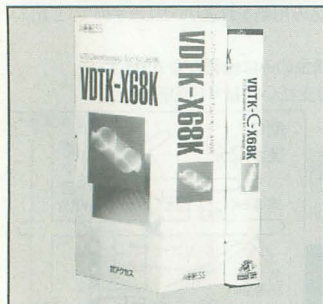
ACCESS

for X68000

誰も知らなかった

V70アクセラレータがどうして速いのか
そなただけにそっと教えてしんぜよう。
実はな、魔法のランプのように
魔人が住んでいるからなのじゃよ。
この魔人は、X68000の命令なら
何でも言う事を聞くのじゃが
特に芸術的な事ならなんでもござれの
凄いやつじゃ。
グラフィック演算でも、レイトレーシングでも
そなたが命令すればあっという間じゃ!
V70とX68000との面倒なやりとりも
全部やってくれるから
そなたはゆっくり茶でも飲んでいてくだされ。
おっと、そろそろ絵が描き上がる頃じゃ。
このことは決して他の者に教えるでないぞ……。

ヒ・ミ・ツ

V70 アクセラレータ　VDTK-X68K

VDTK-X68Kの仕様

- ●V70 CPU(μPD70632)
  20MHz 32ビットマイクロプロセッサ
- ●V70AFPP(μPD72691)
  フローティング・ポイント・プロセッサ
- ●メインメモリ(DRAM)2Mバイト
  同一ページ内のアクセスはNo Wait
- ●共有メモリ(SRAM)128Kバイト
  X68000との通信用
- ●併行動作　X68000とV70は、併行に動作することが可能。
  データの受け渡し処理のために双方向ハンドシェークI/Oポートを搭載。

※製作：ボード……………有限会社アクセス
　　　ソフトウェア………株式会社ハドソン

同梱ソフトウェア

- ●アセンブラ
- ●リンカ
- ●ソースコードデバッガ
- ●システムモニタ
- ●フロートエミュレータ
- ●コマンドシェル

オプションソフトウェア

- ●Cコンパイラ
  (VDTK-C-X68K)

価格

●ボードパッケージ　(XVI対応)
VDTK-X68K………………………¥248,000

●オプションソフト (Cコンパイラ)
VDTK-C-X68K…………………¥68,000

購入方法

上記商品は当面の間、通信販売のみとさせて頂きます。
購入ご希望の方は、住所、(社名、所属)氏名、電話番号をお知らせ下さい。注文書をお送りいたします。

有限会社アクセス
〒101 東京都千代田区神田神保町1-64
神保町協和ビル7F
☎03(3233)0200(代)　FAX.03(3291)7019

資料請求券
oh!X
4月号

パソコン/ワープロ通信ネットワークサービス

J&P HOT LINE

第3回　クレィジーマミさん　ID：JH001578

今回は、J&P　HOTLINEの「SHARP-HOTLINE」SIG-OP（シグオペ）クレィジーマミさんの登場です。ネットワーク上の出会いから結ばれた奥さまと夫婦そろって積極的にアクセスをされています。そんなネットワーカーの理想像ともいえるクレィジーマミさんに、MZシリーズの魅力をうかがいました。

＝基本データ＝
- 使用機種名：SuperMZ Model 30（MZ-2521）
- 所有周辺機器：MZ-1V01（イメージスキャナ/FAX）
MZ-1X19（モデムフォン）　MZ-1M08（音声合成ボード）
MZ-1E26（ボイスコミュニケーションボード）
- 使用開始時期：1985年冬ごろ
- おすすめMZ用フリーソフト：自分がアイデアを出した「ジャリテン」
自分でつくった「アルゴディスクエディタ」
（いずれもSHARP-HOTLINE「ライブラリーMZ」にあり）

■MZ-2500購入の理由は？

SHARPの製品で、MZシリーズが好きだから。また、完全に日本語に対応した、使いやすい高速なBASICなので。機能も豊富で、まだ使いきれてないくらい。

■MZ-2500のよいところ、楽しい部分は？

今でも十分使える性能。特にBASICは使いやすくて、某国民機でプログラミングするのは苦痛だと感じるほど。アルゴ機能も便利だし楽しめます。また、SHARPは電卓も作っているので、演算の精度が良く、安心して数値計算に使えます。

■MZ-2500の面白い使い方を教えて下さい。

ボイスボードを入れてあるので、キー入力を合成音声で確認できるようにすると、データ打ち込みが楽で高速になります。
なかには、留守番電話とFAX着信の自動切り替えをさせている人もいるそうです。

■MZ、Xユーザーに知らせたいHOTLINEのコーナーは？

SHARP-HOTLINEの「Oh！X68K」のボードでは、PDSのディスク回覧を定期的にやっていますよ。

■あなたにとって、J&P　HOTLINEとは？

朝刊。いつもAPがすいている朝につないでいるから。
いろいろなニュースを知ることも出来るし、心のなかのものを吐き出す時にも使えます。
その上、ぬるめの風呂のように、一旦入るといつまでも入っていたくなる。

■HOTLINEを何に活用されていますか？

新聞をとっていないので、耳寄りな情報のほとんどはSIGの書き込みから得ている。
そして、過去には、花嫁探し／v(^o^)vダイセイコウ

J&P HOT LINEへのご入会はスタータキットで。

お求めは、下記のお店へ。又は現金書留にて、¥3,000＋¥90(消費税3%)＝¥3,090を事務局までお送り下さい。
すぐにスタータキットをお送りします。

お問い合わせは――――
〒556 大阪市浪速区日本橋西1-6-5 上新電機株式会社
J&P HOTLINE事務局宛　TEL.(06)632-2521

スタータキットのお求めはJ&P各店でどうぞ。

店名	住所	電話
渋谷店	東京都渋谷区道玄坂2－28－4	☎(03)3496-4141
町田店	東京都町田市森野1－39－16	☎(0427)23-1313
八王子店	東京都八王子市旭町1-1八王子そごう7F	☎(0426)26-4141
立川店	東京都立川市幸町4－39－1	☎(0425)36-4141
三鷹店	東京都三鷹市野崎1－20－17	☎(0422)31-6251
横浜店	横浜市西区北幸2-9-5横浜HSビル1F	☎(045)313-6711
本厚木店	神奈川県厚木市中町3－4－4	☎(0462)25-5151
津田沼店	千葉県習志野市津田沼1-11-2	☎(0474)72-5211
越谷店	埼玉県越谷市神明町2-87-1	☎(0489)66-1221
焼津インター店	静岡県焼津市越後島385	☎(054)626-3311
富山店	富山市掛尾町300	☎(0764)22-5033
金沢店	金沢市入江2－63	☎(0762)91-1130
寺地店	金沢市寺地2－3	☎(0762)47-2524
大須店	名古屋市中区大須4－2－48	☎(052)262-1141
テクノランド	大阪市浪速区日本橋5－6－7	☎(06) 634-1211
メディアランド	大阪市浪速区日本橋5－8－26	☎(06) 634-1511
コスモランド	大阪市浪速区難波中2－1－17	☎(06) 634-3111
U. S. LAND	大阪市浪速区日本橋4－9－15	☎(06) 634-1411
ビジネスランド	大阪市北区梅田1-1-3大阪駅前第3ビルB2	☎(06) 348-1881
高槻店	高槻市高槻町11－16	☎(0726)85-1212
くずは店	枚方市楠葉花園町15－2	☎(0720)56-8181
千里中央店	豊中市新千里東町1-3 SENCHU PAL 2番街4F	☎(06) 834-4141
摂津富田店	高槻市大畑町24－10	☎(0726)93-7521
寝屋川店	寝屋川市緑町4－20	☎(0720)34-1166
枚方ハイハス店	枚方市田口3－41－7	☎(0720)48-1211
藤井寺店	藤井寺市岡2－1－33	☎(0729)38-2111
岸和田店	岸和田市土生町2451－3	☎(0724)37-1021
さんのみや1ばん館	神戸市中央区八幡通3－2－16	☎(078)231-2111
西宮店	西宮市河原町5－11	☎(0798)71-1171
伊丹店	伊丹市昆陽池1－63	☎(0727)77-5101
姫路店	姫路市東延末1-1住友生命姫路南ビル1F	☎(0792)22-1221
京都寺町店	京都市下京区寺町通仏光寺下ル恵比須之町549	☎(075)341-4411
京都近鉄店	京都市下京区烏丸通七条下ル東塩小路町702	☎(075)341-5769
和歌山店	和歌山市元寺町4－4	☎(0734)28-1441
和歌山南店	和歌山市中島368	☎(0734)25-1414
学園前店	奈良市学園北1－8－10	☎(0742)49-1411
奈良1ばん館	奈良市三条町478－1	☎(0742)27-1111
新大宮店	奈良市法華寺町83－5	☎(0742)35-2611
郡山インター店	大和郡山市横田693－1	☎(07435)9-2221
田原本店	奈良県磯城郡田原本町千代574-1	☎(07443)3-4041
熊本店	熊本市手取本町4－12	☎(096)359-7800

なか身は、どちらも32ビット。

プロセッサの未来を先取、洗練されたアーキテクチャを誇るMPU MC68000シリーズを搭載。
先駆のクリエイティブ・アビリティで使う人の創造性に応える68ワールドへ、どうぞ。

●お問い合わせは…
シャープ株式会社 コンシューマーセンター西日本相談室〒545大阪市阿倍野区長池町22番22号☎(06)621-1221(大代表) 電子機器事業本部システム機器営業部〒545大阪市阿倍野区長池町22番22号☎(06)621-1221(大代表)

T1002179040602 雑誌 02179-4